

Adding Semantics to Social Software Engineering: (Re-)Using Ontologies in a Community-oriented Requirements Engineering Environment

Steffen Lohmann¹ and Thomas Riechert²

¹DEI Laboratory, Computer Science Department
Carlos III University of Madrid, Spain
slohmann@inf.uc3m.es

²Agile Knowledge Engineering and Semantic Web
Department of Computer Science, University of Leipzig, Germany
riechert@informatik.uni-leipzig.de

Abstract: Social Software is typically characterized by low formal semantics and weakly structured contents. Software Engineering, in contrast, requires at least a certain degree of formality and structure. In order to face these opposing demands, we propose to ground Social Software Engineering on concepts and technologies from the Semantic Web. In particular, we present a Semantic Wiki based approach in this paper that uses well-known ontologies to represent activities and artifacts that emerge in an online environment for community-oriented Requirements Engineering. We illustrate the benefits of reusing these domain-independent ontologies and show how they can fruitfully be combined and interlinked in one common upper ontology.

1 Introduction

The development of software is usually a highly social activity that involves people with diverse backgrounds and from different domains. The crucial role of the ‘social aspects’ of Software Engineering is well-known for a long time but it receives renewed attention these days. The new interest is mainly driven by recent phenomena in the context of online communities and social interaction on the web that are often subsumed under terms such as “Social Software”, “Web 2.0”, or “Social Web” [Por08]. Understanding these phenomena and their underlying principles and success factors and exploiting them to improve the support for social interaction in Software Engineering is currently a main goal in several research efforts and topic in a number of workshops.¹

In this context, the term “Social Software Engineering” (SSE) has been coined to emphasize the importance of social aspects in software development and to investigate the

¹To name the acronyms of just a few workshops from the last two years that focused on social interaction in software development: SSE at SE '10, CHASE at ICSE '09 & '08, SER 2.0 at CASCON '09, SENSE at SE '09, SofTEAM at SE '09, SoSEA at ESEC/FSE '09 & ASE '08.

application of concepts, principles, and technologies from the domain of Social Software to Software Engineering.² Although it is difficult to give an exact definition for “Social Software Engineering”, two aspects are central to this understanding of software development: It is focused on lightweight, community-centered collaboration and uses online environments to share artifacts and knowledge about a software product. These online environments are either accessible via a web browser (such as a web application or website) or integrated in the participants’ working environments (such as collaboration features in an Integrated Development Environment (IDE)).³

However, in contrast to Social Software that is characterized by ad-hoc collaboration, weakly structured contents, and low formal semantics, qualitative Software Engineering usually demands structured processes, well-defined artifacts, and high formal semantics. Even though several agile development methods emerged in recent years that are build around the principles of self-organization, frequent interaction, and simplicity [Coc01], at least a certain degree of formality and structure is unavoidable for successful software development. Balancing these opposing demands is a major challenge in Social Software Engineering, as we already pointed out in [LDHH09].

We see a possible solution approach in the application of structured representation formats from the Semantic Web. As we try to show in this paper, ontologies already available in the Semantic Web can be highly suitable to represent collaboratively created artifacts, involved participants, and (self-)organizing structures emerging in online environments for Social Software Engineering.

After a brief introduction into the general topic of applying ontologies in Software Engineering in Sec. 2, we demonstrate how widely adopted ontologies of the Semantic Web can be adequately (re-)used as conceptual grounding in a web environment for community-oriented Requirements Engineering in Sec. 3. We highlight the benefits and possibilities that result from the integration of these ontologies and describe how we combined them in one common upper ontology. Finally, we summarize and discuss the benefits and limitations and draw some general conclusions in Sec. 4.

2 Ontologies in Software Engineering

The idea of applying ontologies in Software Engineering is not a new one. Many attempts have been made in the last few years to integrate ontological and software engineering.⁴

²The term “Social Software Engineering” is sometimes also used to describe the “engineering of social software”. However, this interpretation of the term is not in the focus of the paper.

³Similar to the nuanced differences between the meanings of the terms “cooperation” and “collaboration” [MMCM92] there exist nuanced differences between the meanings of “collaboration” and “social interaction” regarding online communities such that “social interaction” is commonly seen as being more ad-hoc, intrinsic, and hedonic but less pre-determined and goal-oriented than “collaboration”. However, these nuanced differences in term meanings shall not be further discussed or elaborated in this paper.

⁴A very popular and condensed definition of the term “ontology” as used in computer science is given with “a specification of a conceptualization” [Gru93]. The main difference between ontologies and conceptual models (such as entity-relationship or UML models) in Software Engineering is often seen in their different scope. Whereas the life-cycle of a model typically ends with a particular project, ontologies are usually valid for several

A popular example is OMG's "Ontology Definition Metamodel (ODM)" [Obj09] specification which defines a family of metamodels for mappings between language constructs of the Semantic Web (such as RDF and OWL) and Software Engineering (such as UML and MOF). Further ideas are expressed in the "Ontology Driven Architecture (ODA)" [TPO⁺06] note and the "Semantic Web Primer for Object-Oriented Software Developers" [KOTW06] of the W3C. But these are only a small portion of the many approaches that have been proposed in this field of research.

An overview on the different application areas of ontologies in Software Engineering is given by Happel and Seedorf [HS06]. They distinguish between the usage of ontologies at development and run-time. Examples for the first category are conceptual models of the problem domain, formal descriptions of components and other software artifacts, or definitions of mappings between different modeling languages (such as proposed by the ODM). At run-time, ontologies are used as part of the application logic or as a conceptual layer that supports dynamic system behavior such as adaptations or automatic discovery and composition of Web Services. Furthermore, they differ between approaches that use ontologies to model a system's context or domain and approaches where ontologies are part of the system or development infrastructure itself.

According to this categorization, our approach is best described as what Happel and Seedorf call "Ontology-enabled development (OED)" [HS06], since ontologies are used at development time to support community-oriented Requirements Engineering.

2.1 Related Work

Closely related to our work is research on the application of Semantic Wikis in Software Engineering. Since Semantic Wikis tackle the aforementioned problem of offering an "architecture for participation" by simultaneously maintaining structure and formality⁵, they can be considered the ideal tool for Social Software Engineering. However, this research area is rather young and there exist only few approaches that try to apply Semantic Wikis in Software Engineering so far.⁶

Early work has been evolved in the context of the RISE project: Decker et al. proposed the idea of "self-organized reuse" in which software artifacts are collaboratively created and structured in a Semantic Wiki [DRR⁺05]. An ontology is semi-automatically derived from these structured Wiki contents and can be reused in related projects. It can also be used for reasoning to enable recommendations and consistency checks. Decker et al. illustrate the general applicability of their approach with an example from Requirements Engineering

projects, as they describe a specific domain, broader application area, or some more general knowledge facts (cp. [MPH08]).

⁵"Semantic Wikis try to combine the strengths of Semantic Web (machine processable, data integration, complex queries) and Wiki (easy to use and contribute, strongly interconnected, collaborativeness) technologies", according to an understanding of "The Semantic Wiki Community" as expressed at <http://www.semwiki.org> at time of writing.

⁶An introduction into Semantic Wikis and their potentials to support knowledge management in Software Engineering is given in [MPH08].

and propose a basic ontology that defines initial templates for five document types. However, their descriptions remain on a conceptual level; a concrete implementation within a Semantic Wiki or a running prototype is not presented.

A prototypical implementation of a Semantic Wiki for Software Engineering is presented by Happel and Seedorf with their Ontobrowse system [HS07]. Ontobrowse adopts the Semantic Wiki paradigm to share knowledge about software architectures, in particular Service-oriented Architectures (SOAs). The system uses the Web Ontology Language (OWL) to represent architectural descriptions and defines an initial ontology for documenting SOAs that can easily be adapted and extended by a responsible admin.

2.2 Discussion of Related Work

Although several approaches have been proposed that apply ontologies in software development, we are not aware of any attempt that systematically reuses and combines more general ontologies from the Semantic Web. Existing approaches either define their own ontologies, reuse software-related ontologies, or provide a framework or infrastructure that assists in the development of ontologies as software artifacts. However, many activities and artifacts are not specific to Software Engineering but can also be found in the same or a similar form in other contexts. This is also and especially the case for community-oriented communication and collaboration that is similarly implemented in various online environments following reoccurring patterns [SL07].

Many widely adopted ontologies exist already in the Semantic Web that are suitable to represent these reoccurring patterns and can therefore be fruitfully reused in Software Engineering. Such a cross-domain reuse has several benefits as we aim to illustrate in the following by an example of Social Software Engineering.

3 Ontologies for Community-oriented Requirements Engineering

We developed a web environment for community-oriented Requirements Engineering within the SoftWiki project [Sof09] that borrows several Social Software concepts to support lightweight collaboration among geographically distributed stakeholders [LDHH09]. It aims to foster a more direct engagement of larger groups of stakeholders in the collection, discussion, development, and structuring of software requirements and focuses on early phases of Requirements Engineering. Especially stakeholders that are not familiar with formal Requirements Engineering and corresponding tools are enabled to express their ideas, wishes, and requirements regarding a planned software product.⁷

The web environment is based on OntoWiki, a comparatively mature and feature-rich Semantic Wiki that has been considered particularly suitable for application in Software En-

⁷Please refer to [LDHH09] for a detailed description of the collaborative features that are provided by the web environment.

engineering [MPH08, LRA08]. Figure 1 shows a screenshot of the web environment that is conceptually divided into its different user interface parts and annotated with some of the ontology concepts used to represent the corresponding artifacts and information pieces. The web environment has been realized as an OntoWiki plugin and the applied ontologies are all imported and integrated in one common upper ontology that is based on the Semantic Web description languages RDF(S) and OWL [AH08].⁸

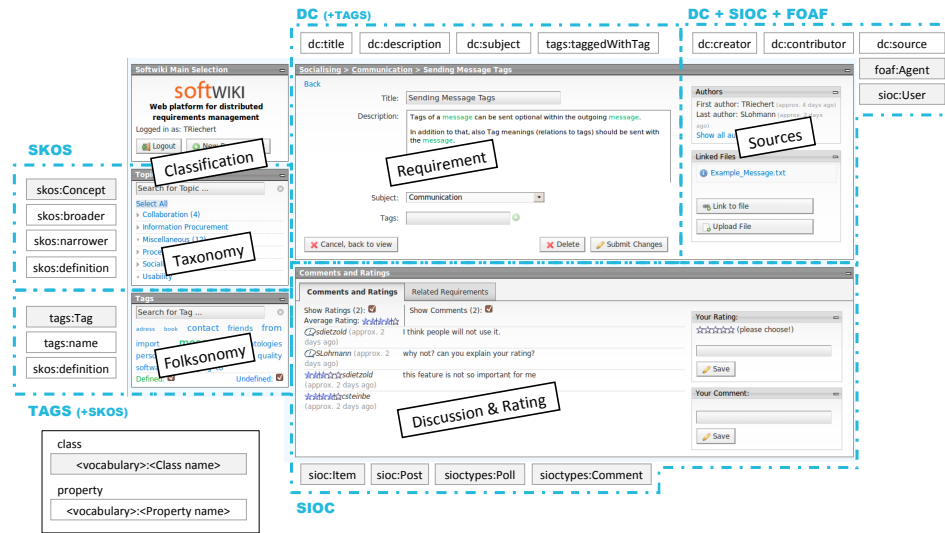


Figure 1: Screenshot of the web environment for community-oriented Requirements Engineering; divided into its conceptual structure and annotated with some of the used ontology concepts.

3.1 Representation of Requirements Metadata via Dublin Core

Following the idea of community-oriented Requirements Engineering, requirements are usually collaboratively edited by all registered users of the web environment [LDHH09]. Basically, a single requirement can be seen as a typical information resource with well-known properties, such as a title, a description, and one or more authors that created the requirement. A set of fifteen often used properties for the description of information resources is defined by the “Dublin Core Metadata Element Set” [Dub08]. We used several of these properties to semantically annotate requirements in our web environment, namely title, description, creator, contributor, subject, and source.

Because such metadata is often valuable in the analysis, refinement, and prioritization of requirements (e.g., to get back to the author(s) or document source of a requirement), maintaining it across different tools is usually of high interest. For that reason, common requirements exchange formats include some metadata elements (e.g., RIF [WH05]).

⁸A demo installation of the web environment can be tested online at: <http://softwiki.de/demo/english>

However, using Dublin Core instead to describe basic metadata extends the range of tools that can be used for accessing and refining the requirements to all tools that are capable to read and interpret this standard, including non-CASE tools. For instance, sophisticated word processing software might be used to revise and spell-check requirements without a loss of crucial metadata.

With Dublin Core, it is also possible to represent basic information for revision control in order to maintain traceability in certain cases (e.g., in collaborative editing). Of course, the metadata properties defined by Dublin Core are not sufficient to represent all information that might be captured about a requirement in all possible cases. But they provide at least a major metadata subset that can be valuably reused in Software Engineering.

3.2 Classification of Requirements via SKOS and TAGS

The web environment offers two ways of classifying requirements: They can either be assigned to a concept of a pre-defined taxonomy or be enriched with an arbitrary number of freely chosen keywords (so-called “tags”). These tags are aggregated to a “folksonomy” [SCH08] that is visualized as “tag cloud” [SCH08] in the user interface beneath the taxonomy tree (see Fig. 1) and can be used for filtering and navigation [LDHH09].

The taxonomy is represented via the “Simple Knowledge Organization System (SKOS)” [MB09]. SKOS has been developed specifically for the definition of controlled vocabularies and is – such as Dublin Core – formally described in an RDF schema that we import. In concrete, we use the `broader` and `narrower` properties to represent the hierarchical taxonomy structure and the class `Definition` to add definitions to concepts. SKOS allows to represent the taxonomy separately from its requirements what eases its export and import. This idea is similar to that of related work [DRR⁺05, HS07, LRA08], except that a standardized vocabulary like SKOS extends the range of applications and projects that can easily work with the taxonomies.

The tags of the folksonomy are represented by the “TAGS ontology” [New05]. Since we defined `Tag` to be a subclass of `SKOS Concept`, a transformation between both is easily possible. For instance, selected tags might be enriched with a definition and/or added to the taxonomy if desired [LDHH09].

3.3 Representation of Stakeholders and Discussions via FOAF and SIOC

Ontologies are also applied to represent the stakeholders that use the web environment. Here, we imported the FOAF (“Friend of a Friend”) vocabulary that allows the representation of persons and their interrelations [BM10]. In particular, we use the FOAF class `Agent` and its subclasses `Person`, `Group`, and `Organization`. However, further FOAF concepts might be linked if desired in order to express additional information about stakeholders, such as contact details or projects the stakeholders are assigned to. Furthermore, the FOAF structure can easily be exported and visualized as social graph with

appropriate tools. This enables new possibilities for social network analysis that might be particularly useful for Social Software Engineering [KHA09].

In addition, the web environment provides features for the discussion of requirements (see Fig. 1) as an important part of community-oriented Requirements Engineering [LDHH09]. We imported the SIOC ontology in order to represent these online discussions. A stakeholder's account is represented by the SIOC class `User`, comments are a subclass of `Post` and ratings are represented by the class `Poll`. This formal representation of ratings via SIOC opens up new possibilities for the prioritization of requirements. For instance, automatic reasoning can support the calculation of priority values based on the ratings. Algorithms can also weight requirements differently, depending on information about stakeholders that is provided by FOAF and SIOC.

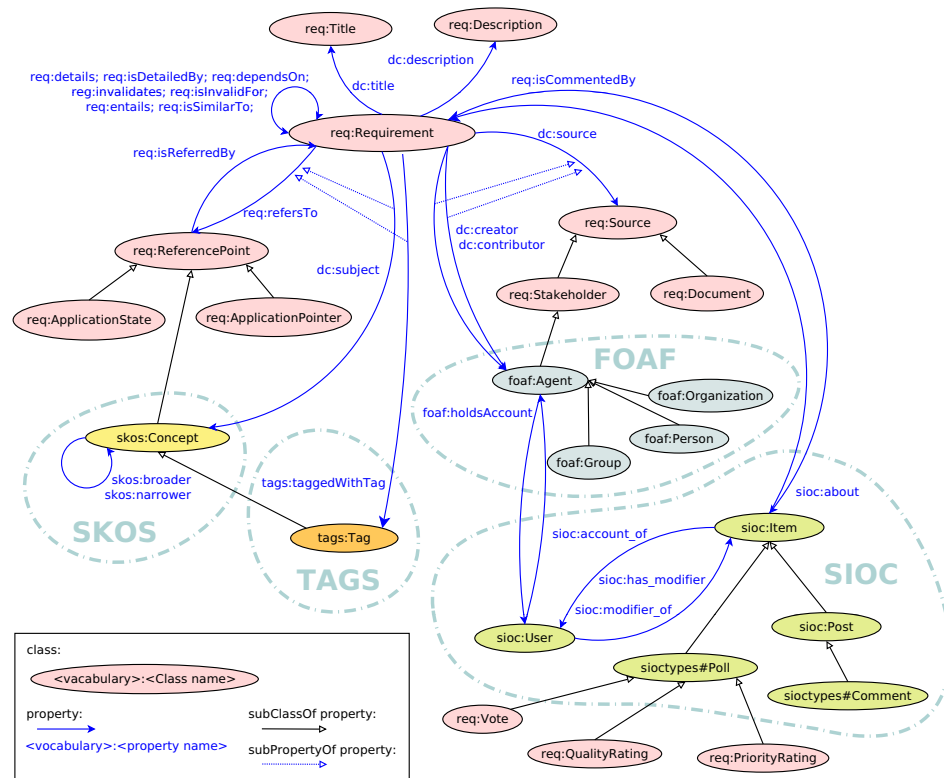


Figure 2: Visualization of the main classes and properties of the SWORE upper ontology with integrated and aligned Semantic Web ontologies.

3.4 Integration in the Upper Ontology SWORE

All ontologies were integrated in one common upper ontology that describes their interrelations. The basic building blocks of Requirements Engineering are represented by the three top-level classes `Requirement`, `Source`, and `Reference Point`: Requirements are derived from sources (e.g., stakeholders or documents) and are classified in a certain way (e.g., by adding a concept or tag). The basic structure of this upper ontology (that we call SWORE) is visualized in Fig. 2.⁹

In addition, the SWORE ontology defines a number of domain-specific concepts that are not covered by the imported ontologies but required to represent the implemented features of community-oriented Requirements Engineering (these elements are marked with the namespace `req` in Fig. 2). For instance, a set of pre-defined relation types is provided that describes dependencies between requirements, such as `details`, `entails`, or `invalidates`. It also supports the definition of pointers to parts of a system or prototype if requirements should be linked with these in certain cases [LR08]. Furthermore, it defines sub-types for ratings that allow an advanced reviewing and prioritization of requirements.

The web environment provides a REST interface that enables access to the structured requirements data in the RDF querying language SPARQL. Instance data of SWORE and all integrated ontologies can also be exported and imported in RDF format. This flexible access opens up a large range of application scenarios. As discussed above, it especially supports the utilization of the requirements data in other tools by maintaining semantic interoperability.

4 Discussion

Since Software Engineering is highly interdisciplinary, many of its aspects – especially domain-independent ones – are often already formally described by well-designed ontologies of the Semantic Web. Reusing these ontologies can be valuable, as we tried to point out in this paper. We presented an online environment for community-oriented Requirements Engineering as illustration and proof-of-concept, but the general idea is also applicable to other parts of Software Engineering.

A key benefit of this cross-domain reuse of ontologies is the resulting interoperability with further tools, including tools that have not been developed specifically for Software Engineering. This opens up new opportunities to utilize, enhance, and analyze artifacts and metadata. We addressed some examples in this paper, such as an external refinement of the classification structure or an advanced analysis of the social network. The modular structure of the SWORE ontology with its clear conceptual separation additionally facilitates access to single, integrated ontologies. However, due to the ontologies' high degree of self-description (e.g., typed links, class and property descriptions, etc.), all instance data

⁹An earlier version of SWORE (without integrated ontologies) is introduced in [RLL07]. The current version is available at: <http://softwiki.de/swore>

can usually also be accessed by more generic Semantic Web tools (e.g., ontology editors, inference engines, etc.) and further processed in various ways, including reasoning and consistency checking.

However, also from a non-technical and non-tool-oriented perspective can the presented approach of ontology reuse be valuable, since it fosters a shared understanding and common practices that are based on elaborated knowledge. Ontologies are normally developed by experts in a field; reusing these ontologies can therefore avoid redundant modeling effort and lowers the risk for wrong interpretations and misconceptions.

Of course, not all aspects of software development can be captured by available ontologies. Additional modeling is usually required if a large part of the collected artifacts and metadata should be formally represented, as in case of our community-oriented Requirements Engineering approach. Furthermore, grounding Software Engineering completely on ontologies raises issues of performance and scalability. Although large improvements have been made in this area recently, short response times and fluent interaction on large-scale datasets are often still a problem with the description languages of the Semantic Web [BS09].

The SWORE ontology describes only a small subset of the many aspects that are part of Requirements Engineering. However, our goal was not a comprehensive domain description but rather the development of an ontology that formally represents artifacts emerging in an online environment for community-oriented Requirements Engineering. SWORE might nevertheless be a good starting point for the development of a more comprehensive ontology for Requirements Engineering. Thus, future work includes an extension of SWORE by integrating further Semantic Web ontologies where applicable and defining domain-specific concepts where not.

References

- [AH08] D. Allemang and J. Hendler. *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Morgan Kaufmann, 2008.
- [BM10] D. Brickley and L. Miller. FOAF Vocabulary Specification, Version 0.97. <http://xmlns.com/foaf/spec/>, 2010.
- [BS09] C. Bizer and A. Schultz. The Berlin SPARQL Benchmark. *International Journal of Semantic Web and Information Systems*, 5(2):1–24, 2009.
- [Coc01] A. Cockburn. *Agile Software Development: Software Through People*. Addison-Wesley Longman, 2001.
- [DRR⁺05] B. Decker, E. Ras, J. Rech, B. Klein, and C. Hoecht. Self-organized Reuse of Software Engineering Knowledge Supported by Semantic Wikis. In *Proc. of SWESE'05*, 2005.
- [Dub08] Dublin Core Metadata Initiative. Dublin Core Metadata Element Set, Version 1.1. <http://www.omg.org/spec/ODM/1.0>, 2008.
- [Gru93] T.R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

- [HS06] H.-J. Happel and S. Seedorf. Applications of Ontologies in Software Engineering. In *Proc. of SWESE'06*, 2006.
- [HS07] H.-J. Happel and S. Seedorf. Ontobrowse: A Semantic Wiki for Sharing Knowledge about Software Architectures. In *Proc. of SEKE'07*, pages 506–512. W3C, 2007.
- [KHA09] T. Kramer, T. Hildenbrand, and T. Acker. Enabling Social Network Analysis in Distributed Collaborative Software Development. In *Software Engineering 2009 – Workshopband*, pages 255–266. GI, 2009.
- [KOTW06] H. Knublauch, D. Oberle, P. Tetlow, and E. Wallace. A Semantic Web Primer for Object-Oriented Software Developers. <http://www.w3.org/TR/sw-oosd-primer/>, 2006.
- [LDHH09] S. Lohmann, S. Dietzold, P. Heim, and N. Heino. A Web Platform for Social Requirements Engineering. In *Software Engineering 2009 – Workshopband*, pages 309–315. GI, 2009.
- [LR08] S. Lohmann and A. Rashid. Fostering Remote User Participation and Integration of User Feedback into Software Development. In *Proc. of I-USED'08*, 2008.
- [LRA08] S. Lohmann, T. Riechert, and S. Auer. Collaborative Development of Knowledge Bases in Distributed Requirements Elicitation. In *Software Engineering 2008 – Workshopband*, pages 22–28. GI, 2008.
- [MB09] A. Miles and S. Bechhofer. SKOS Simple Knowledge Organization System Reference. <http://www.w3.org/TR/2009/REC-skos-reference-20090818/>, 2009.
- [MMCM92] P.W. Mattessich, M. Murray-Close, and B.R. Monsey. *Collaboration: What Makes It Work*. Amherst H. Wilder Foundation, 1992.
- [MPH08] W. Maalej, D. Panagiotou, and H.-J. Happel. Towards Effective Management of Software Knowledge Exploiting the Semantic Wiki Paradigm. In *Software Engineering 2008*, pages 183–197. GI, 2008.
- [New05] R. Newman. Tag Ontology Design. <http://www.holygoat.co.uk/projects/tags/>, 2005.
- [Obj09] Object Management Group. Ontology Definition Metamodel, Version 1.0. <http://www.omg.org/spec/ODM/1.0>, 2009.
- [Por08] J. Porter. *Designing Social Web Applications*. New Riders, 2008.
- [RLL07] T. Riechert, K. Lauenroth, and J. Lehmann. SWORE - SoftWiki Ontology for Requirements Engineering. In *Proc. of CSSW'07*, pages 111–118. GI, 2007.
- [SCH08] J. Sinclair and M. Cardew-Hall. The Folksonomy Tag Cloud: When is it Useful? *Journal of Information Science*, 34(1):15–29, 2008.
- [SL07] T. Schümmer and S. Lukosch. *Patterns for Computer-Mediated Interaction*. John Wiley & Sons, 2007.
- [Sof09] SoftWiki: Distributed, End-user Centered Requirements Engineering for Evolutionary Software Development. <http://softwiki.de/netzwerk/en/>, 2009.
- [TPO⁺06] P. Tetlow, J. Pan, D. Oberle, E. Wallace, M. Uschold, and E. Kendall. Ontology Driven Architectures and Potential Uses of the Semantic Web in Systems and Software Engineering. <http://www.w3.org/2001/sw/BestPractices/SE/ODA/060103/>, 2006.
- [WH05] R. Wiebel and S. Höh. Requirements Interchange Format (RIF), Version 1.0. <http://www.automotive-his.de/rif/>, 2005.