

iCamp

< D3.3 >

An Interoperability Infrastructure for Distributed Feed Networks

Former Title: Specification of an Application Programming Interface for Distributed Collaboration and Social Instruction and Prototype of Collaboration Network

Bibliographic Details.

Deliverable Number:	D3.3
Contractual Date of Delivery:	03 / 2007
Actual Date of Delivery:	07 / 2007
Work Package(s) Contributing:	WP3
Nature of the Deliverable:	Report / Prototype
Status:	Final
Security (Distribution Level):	Public
Editor(s):	Fridolin Wild, VUE
Project start:	October 1 st , 2005
Duration:	36 months, STREP / IST

Amendment History.

Version	Date	Contributor(s)	Modification
0.0.0		Stefan Sobernig, Fridolin Wild	Definition of Job Ticket Weblog Integration.
0.1.0	May 10, 06	Anna Danielewska-Tuńska, Dariusz Górka	Technical Report.
0.1.5	Dec 14, 06	Antonio Tapiador, Stefan Sobernig, Fridolin Wild	Preliminary Review.
0.2.0	Jan 4, 07	Anna Danielewska-Tuńska, Dariusz Górka	First Raw Version.
0.2.5	Jan 4, 07	Fridolin Wild	Review Raw Version.
0.3.0	Jan 21, 07	Anna Danielewska-Tuńska, Dariusz Górka	Second Revision.
0.4.0	Feb 8, 07	Fridolin Wild	Consolidated Version, Restructuring, Introduction, Problem Statement.
0.5.0	Feb 14, 07	Fridolin Wild, Antonio Tapiador	Review Second Revision.
0.6.0	Feb 16, 07	Antonio Tapiador, Dariusz Górka, Fridolin Wild	Work Distribution Prototype (Darek), Background (Antonio), Concept Space (Fridolin).
0.6.1	April 17, 07	Fridolin Wild, Steinn Sig- urdarson, Stefan Sobernig	Pingback Strategy, Scenarios (not yet integrated), Code Review Moodle / Wordpress.
0.6.5	April 19, 07	Fridolin Wild	Fitted into template, Reorganised, Scenarios added.
0.6.6	May 07, 07	Fridolin Wild	Scenarios elaborated, Specification rewritten.
0.6.9	May 09, 07	Fridolin Wild	Specification amended, Contributions of partners added.
0.7.0	June 12, 07	Fridolin Wild	Consolidation.
0.7.1	June 20, 07	Fridolin Wild	Section Interoperability Rewritten, Section Motivation enhanced, Reference Model finalised, Updated Interaction Standards.
0.7.2	June 21, 07	Fridolin Wild	Standards Section enhanced (content standards, interaction standards).
0.7.3	June 22, 07	Fridolin Wild	Software products section enhanced.
0.7.4	June 25, 07	Fridolin Wild	Authorisation / Authentication section reworked.
0.7.7	June 26, 07	Fridolin Wild	Blogger API, OPML added.
0.8.2	June 27, 07	Fridolin Wild	API overview, blog applications overview reworked.
0.8.3	June 28, 07	Fridolin Wild	API functionality aggregation added.
0.9.0	June 29, 07	Fridolin Wild	LMS application review, aggregator review.
0.9.3	June 30, 07	Fridolin Wild, Ahmet Soylu, Steinn Sigurdarson	Snapshots added.
0.9.4	June 30, 07	Fridolin Wild	Validity Check: Description.
0.9.5	July 3, 07	Fridolin Wild	Validity Check: Description finalised.
0.9.7	July 4, 07	Fridolin Wild, Anna Danielweska-Tuńska	References checked.
0.9.8	July 5, 07	Fridolin Wild	Conclusion for related work section added.
1.0.0	July 5, 07	Fridolin Wild	Typesetting.

Editor

Fridolin Wild (VUE)

< D3.3 >

An Interoperability Infrastructure for Distributed Feed Networks

Contributors

Fridolin Wild (VUE), Steinn Sigurdarson (VUE),
Stefan Sobernig (VUE), Ahmet Soylu (ISIK),
Vahur Rebas (TLU), Dariusz Górkka (AGH),
Anna Danielewska-Tułecka (AGH),
Antonio Tapiador (UPM)

Table of Contents

0. Executive summary	9
1. Motivation	11
2. Scenarios	14
2.1. Scenario 1: Simple Feed Management Scenario	14
2.2. Scenario 2: Rip, Mix, and Feed	15
2.3. Scenario 3: Group Blog	15
2.4. Scenario 4: Learning Contract Tool	16
3. Loosely-Coupling for Blogs	17
4. Related Work	20
4.1. Feed Standards	20
4.2. Blog Interaction Standards	22
4.3. Existing Blogging Tools	26
4.4. Current Shortcomings	30
5. The Missing Link: The FeedBack Specification	31
5.1. Communication Process	32
5.1.1. Advertise Subscription Opportunity	32
5.1.2. Acknowledgement of Subscription	32
5.1.3. Inform about Updates	33
5.2. Validity Check: Reference Implementation	34
6. Discussion of Effects & Future Work	38
7. References	39

Executive summary

Blogs – the most popular systems for authoring and managing micro-content – have the affordance to become an integral part of teaching and learning processes as a vehicle for knowledge management. Open, flexible systems integrating blogs provide user-friendly, personalised microlearning environments while ensuring ubiquitous access. The iCamp project aims at creating an infrastructure for collaboration and networking across systems, countries, and disciplines in Higher Education. This virtual environment is composed of various interoperable tools and platforms. One of the key elements, a so-called building block, in this entirely heterogeneous space is a blogging tool.

This paper concentrates on blog interoperability in the context of creating such an open space for learning and collaboration. As blogging is one of the most popular web-based forms of publishing today, there is a plethora of different blogging tools, feed readers, and aggregators, enabling information dissemination, filtering, and retrieval. Several different data structure and interaction standards emerged which makes integration a real challenge. This paper presents the possibility of creating distributed microlearning environments basing on networks of integrated blogs, discusses problems of such an integration along with possible solutions, and proposes an architecture for loosely-coupled blog integration. Furthermore, it documents the prototype of this infrastructure.

The core of this infrastructure consists of an application programming interface (API) for feed management that allows learners as well as facilitators to automatically set up channel structures for feed syndication and effectively reducing the management efforts thereof through proper system support. These channels and the management thereof is common to all scenarios which have been elaborated in the second section: they are necessary to initiate subscription procedures that allow to share and communicate about feeds sources ('simple feed management scenario'); they form the basis for rip, mix, and feed activities; they are a way how to (technically) realise group blogs; and finally they are a necessary prerequisite when realising learning contracts and records of action in a distributed learning environment.

The originally planned title of this deliverable was 'Specification of Application Programming Interface for Distributed Collaboration and Social Instruction and Prototype of Collaboration Network'. As the proposed infrastructure is capable of serving many more purposes besides blog integration – as long as the data to be syndicated can be wrapped into common feed standards as RSS 2.0 or ATOM –, the authors have decided to rename the deliverable to 'An Interoperability Infrastructure for Distributed Feed Networks'.

1. Motivation

The conglomerate of all blogs available online, the so-called 'blogosphere', has been certified to show a bursty evolution at least since 2001 (Kumar et al., 2003), where an eruptive rise can be identified not only regarding metrics of scale but also with respect to deepening community structures and higher degrees of connectedness. As with July 2007, for example, Technorati is indexing 90 million blogs (Technorati, 2007): Blogging is obviously an increasingly popular phenomenon, although meta-studies reveal that between one half and two thirds of all blogs are abandoned within only two months after their creation (Gurzick and Lutters, 2006).

One of the reasons that blogs became so attractive is their ease of use, removing barriers of technoliteracy from web self-publishing (Tepper, 2003). There is a plethora of web-publishing tools, allowing the user to choose from a large variety of (non-) commercial hosting services (often available free of charge). Moreover, users can set up their own web-applications choosing from a rich portfolio of open- and closed-source products. Learning Light's eLearning Centre, for example, lists already back in 2006 more than 56 different products and online services in a vendor directory for blogging tools (eLearning Centre, 2006). Publishing rich content with weblogs does not require any profound technical knowledge – knowledge such as language skills in the hypertext mark-up language required to create pages with a desktop html-editor and skills such as those which are necessary to set up a fully-fledged content management system.

It is not surprising then, that blogs became vehicles for knowledge management to already often form an integral part of teaching and learning processes. Blogs can be used to organize lectures, seminars, and discussions both between teachers and students. Herring et al. (2004) found in their study on blog genres that 57.5% of the blogs' authors investigated in a random sample of 203 blogs are students on a secondary and tertiary level. However, at the same time, the major share of 70.4 % of the blogs are personal journals reporting on the lives of their authors, clearly less are deployed for filtering, i.e. commenting on external content, and knowledge sharing. Similar results are reported by Schmidt & Mayer (2007) in their end-2005 study among German speaking bloggers: users in education (pupils, students) are underrepresented among the blog writers with the (primal) aim for knowledge sharing (a.k.a. knowledge-bloggers = k-loggers), the major share of k-loggers stems from a work context.

The reasons why people create and maintain blogs vary to a large extent, however, always also including community building and social networking among

the key motivations substantiated through empirical studies (Nardi et al., 2005; Schiano et al., 2004). Besides the obvious – group blogs –, community structures and social networks were proven to exist between individual, but networked blogs (Chin and Chignell, 2006).

However, when deploying blogs in collaboration, many obstacles can be found that have not been solved so far. To facilitate “productive blog conversations” which are necessary in knowledge management and learning, “more carefully tailored socio-technical systems are needed”, as De Moor and Efimova (2004) claim. De Moor and Efimova identify the problem of notorious fragmentation of conversations to be responsible for (even the authors’) difficulties in reconstructing conversations. Furthermore, they see (initial) response times as a problem that may slow down conversations, especially when comparing to push technologies such as mailing lists. Another obstacle to productive conversations in collaboration processes is identified to be the low number of links to blog posts, “lower than often expected” (De Moor and Efimova, 2004). Only 51.2 % of all blogs (Herring et al., 2004) link to other blogs, only 53.7 % link to other websites. 30.5 % of all blogs do not link to anything at all (besides badges). The average entry received .3 comments, the majority of entries received none. Multimodality poses yet another problem: replies and comments are often distributed across comment fields, but can also be found in blogs of the repliers. Krause (2004) identifies the fuzziness of the audience as a problem that may be responsible for failing discussion in his course experiment (his article is titled ‘When Blogging Goes Bad’), as it is unclear whether the desired audience (the course participants) will be reached in time and at all.

Looking more closely at why weblogs ‘die’, the general reasons for their abandonment differ for each phase of a blog’s life-cycle. Especially in more mature phases, where revisions of tool choices are not so likely to be found, external causes are prevalent including changes in the social network of the author, e.g., changes in employment or family (Gurzick and Lutters, 2006).

In traditional online media such as organisational discussion boards, as Farmer puts it, users are forced to ‘re-invent’ themselves in each new online work context, as there is “little or no capacity for the development and transference of an online persona from one context to the next” (Farmer, 2005). Ownership and control over communicative artefacts remain with the system providers. Contrarily, blog-based technologies in principle, says Farmer, bear the potential to solve this shortcoming by focusing on individual publishing and flexible aggregation ensuring at the same time that control over information exchange and communication (as a basis for collaboration) resides with the users.

Pragmatically, De Moor and Efimova (2004) call for ways to represent the semantics and pragmatics of the blogosphere (e.g., visualisations) and “methods for gradually developing the required complex socio-technical systems around

blogs, such as virtual communities” (i.e., collaborative management methods). However, we are not quite there: Interoperability between weblogs is not given. Interoperability is defined to be a property that emerges, when distinctive information systems (subsystems) cooperatively exchange data in such a way that they facilitate the successful accomplishment of an overarching task (Wild & Sobernig, 2006). Interoperability therefore requires the participating information systems not only to integrate on a communication and control layer specifying joint communication interfaces and protocols, not only to mutually share a fair amount of knowledge on their (meta-) data schemes and access options, but also to cooperate regarding application semantics and functionality (see Kosanke, 2005).

According to (Pilgrim, 2004), there are nine different and incompatible versions of RSS. Besides problems of being well-formed, several other incompatible content-model changes have been applied throughout the version history of RSS. Furthermore, switching the abstract data-model between RDF and XML caused further confusion. Additionally, ATOM entered the field in 2003 competing with RSS (cf. Wittenbrink, 2005). Looking at interaction protocols, a similar picture can be assessed: besides a publish & subscribe pull-mode via HTTP, a set of APIs to implement a push-mode for data exchange causes simple syndication to become suddenly not so simple anymore. Access restriction and protection virtually does not exist, same applies for application functionality as well as semantics; task support is restricted to simple publishing features and varies from system to system.

In Section 2, we will compose scenarios for interoperability of distributed weblogs in collaborative learning situations in technology-enhanced learning to work out those requirements a reference model for distributed weblog interoperability needs to fulfil.

In Section 3, we are giving a short introduction into theoretic interoperability shortcomings, as can be derived from system integration research. This section also defines the concept space of the subsequent elaborations.

In Section 4, we will carefully review related work on distributed weblog interoperability. We will conduct this review in three stages. First, we will analyse the existing standards for weblog (meta-) data and interaction protocols in more depth. Second, we will pragmatically investigate existing tools regarding their interoperability capabilities. Third, we will show that this existing work has several shortcomings, especially on the layers of application functionality and semantics, but also regarding (meta-) data standards, and access and interaction protocols.

In Section 5, we construct the missing link, an interoperability model for management of distributed weblog networks. Feasibility of this reference model has

been tested in a prototype implementation which will be reported in the second part of this section. In Section 6, we will discuss the effects of the reference model with respect to the scenario and will identify areas of future work.

2. Scenarios

“Scenarios are devices for improving our perception” (van der Heijden, 1997). By putting a complex set of events and relationships into a story, the problem becomes cognitively manageable and can be better memorized (ditto).

The aim of the scenarios we elaborated and collected below therefore is twofold: first of all, we intend to clarify with them the problem of weblog interoperability in the context of socially networked, self-organised, self-directed collaboration. However, as a secondary goal, it at the same time provides a set of test cases against which any proposed solution to the identified problems can be evaluated to see if it solves all crucial sub-problems.

We restrict ourselves in the following scenarios to the context of blogs in order to have a common starting point and provide vivid examples. However, the infrastructure which will be proposed in this document is by nature capable of working with everything that can be transmitted via, for example, an RSS-feed.

Important elements of virtual collaboration are blogs which imposes on us the need of providing integration between various blogging tools, components, and other blog-based applications.

2.1. Scenario 1: Simple Feed Management Scenario

A student affiliated to ISIK university opens and operates a learning diary in the form of a blog provided to him by a multi-author weblog module integrated into CourseOnline. Upon creation time or at any operative time, the student enters into a joint course which is collaboratively offered by ISIK and AGH Kraków. He is required to keep his learning diary equally visible to fellow students and to the facilitator in Kraków who all are solely active in AGH’s Moodle instance. A similar situation applies for his peer at ISIK university also participating in the joint course. Different to him, however, she runs her own blogging tool and operates her learning diary through a WordPress instance she set-up. In both cases, they make their blogging tool advertise the feed to their fellow users of the Moodle system in Kraków, so these can conveniently decide to subscribe to the blogs from ISIK in order to get regular updates.

Later-on, one of the 'nomad' students at ISIK decides to participate in an exchange programme, so he needs to temporarily move his blog from CourseOnline to AGH's Moodle instance. To do so, he creates a feed offer from CourseOnline to be shown to his account in Moodle or directly enters the feed address into Moodle, in order to fetch all his entries and have them at hand in the other system.

2.2. Scenario 2: Rip, Mix, and Feed

A facilitator points all his students to the blog of Greenpeace which they should subscribe to in order to get regular updates about Greenpeace's activities as this is important for his course about 'non-profit organisations'. Furthermore, the facilitator creates automatic advertisements of some tutor's feeds. In some cases he does not know the addresses of the students' learning diaries, so he requests from his students to send him a subscription offer. He receives a bunch of subscription offers, categorizes them with the course related tag 'NPO-Summer-07' and adds them to his feed aggregator. He thereby has his system let the students' blogging systems know that they should inform about updates in the future. Whenever a student is creating, updating, or deleting a posting, their blog automatically informs the facilitator's aggregator to process the changes.

In some cases, when reading his students blog entries, he wants to carry things back to a group discussion, so he offers a facilitator feed where he forwards some of the blog entries every once in a while and comments them. Additionally, he aggregates all feeds of all students together into one feed to easily synchronise the entries into other applications (e.g., his Mozilla RSS feedreader plug-in).

2.3. Scenario 3: Group Blog

When collaborating in small project-groups, students (and eventually also the facilitators) need a common communication channel which they can use to report about progress, rise questions, help each other, etc. An innovative and very flexible approach how to realise this in an interoperable way, is to create a distributed group blog.

A distributed group blog is characterized the way that every participant of this group blog is able to write postings to it in her own learning tool of choice and the postings are subsequently shared with the other participants. Every participant of a group blog has equal access rights, i.e. everybody is allowed to create, edit, or delete (=inactivate) postings. To set-up a group blog, one participant initiates the creation of the group blog in one of the participating systems. She specifies which persons are participating in this group blog and which system they are living in. The system is advertising the group blog to the learning tools of the other participants. The other participants will be required to acknowledge

within their own learning tool of choice that they really want to participate in this shared group blog. Magically, the participants of the group blog will see the group blog appear in their system. Whenever one of the group members creates a new posting (or is updating / deleting an existing one), the other participants will immediately see these changes in their own learning tool of choice.

To ensure a high level of usability, it is important to make clear to the participants which postings are part of this group blog and how they can easily make a posting to this common communication channel. Eventually it turns out to be good that the user may decide on whether he wants to see his own postings in the group blog together with the others' postings or separately.

2.4. Scenario 4: Learning Contract Tool

To install scaffoldings for self-organisation and self-direction, learning contracts can be established with the help of a learning contract tool (Harri-Augstein & Webb, 1995). The contract itself is consisting of a couple of 'areas' where you take down notes about, for example, objectives, resources, tasks, and the situative context. They are realised in a wiki-like form in which the contractor can enter the relevant data in several form fields. This contract page in the wiki is advertised to a facilitator, telling his system that he should subscribe to the page to get informed about changes. Whenever a change takes place, an update notification is sent to all listening systems, in this case to the facilitators. Additionally, the learning contract tool contains a section where progress records are stored: learners are requested to regularly write progress notes through-out the life-time of the contract. Progress records are unstructured postings that inform about the work performed. Students can negotiate their learning contract individually with the facilitator. Usually, a learning contract is changed during its life-time several times, after people reflect on progress and contract obligations. In some situations, it might be a deliberate regular revisiting of the contract and subsequent re-negotiating. In other cases, the re-negotiation only takes place when needed and requested.

At the very end of the contract, feedback has to be given by the contractor about the contract and about the progress records. Often this is a small final review report about the course of the contract. In practice this usually is a beautified version of the compiled progress reports and a little bit more.

Every contract is individually negotiated; a contract is not about group work. However, the negotiation and information process can include also other learners besides a facilitator. This then allows for monitoring the development of changes in others' contracts and an investigation of the perspectives of peer learners. There is support for investigating progress records and for analysing the changes of the contract, for example, through a time-line overview.

3. Loosely-Coupling for Blogs

Interoperability is defined to be a property that emerges, when distinctive information systems (subsystems) cooperatively exchange data in such a way that they facilitate the successful accomplishment of an overarching task (Sobernig et al., 2006). Interoperability therefore requires integration on different levels. At first sight, this definition *sensu latu* might appeal to simplify system integration, as it is about connecting systems, aligning functionality and data standards, and supporting joint tasks. Actually, however, system integration problems are very complex. Below we list some of the weblog integration challenges referring to Hohpe's and Woolf's (2003) general proposition.

1. **Ultimate Holisticity:** The communication to be established is not only between two blogging systems, but also touches the surrounding e-learning systems. This has implications on whole system architecture.
2. **Limited Control:** There is often very limited control over the blogging application, for example when integrating a particular learning environment with an independent, packaged blogging application.
3. **Standards Fragmentation:** There is still the lack of interoperability between standards-compliant blogging applications (due to extensions, different interpretations, limited establishment of standards).
4. **Lack of Shared Meaning:** The data can be represented in XML (the lingua franca of system integration), but it is a rather shallow compatibility – the semantics vary tremendously from system to system. There are semantic differences clearly apparent in the existing feed standards (e.g., between RSS 2.0 and Atom).
5. **Speed of Change:** The applications and systems, even the standards are constantly changing – maintaining such mixture of systems can be challenging itself.

One solution to these problems can be found in the architectural principle of loosely-coupling. This approach leads to more stable and flexible applications. The principle is „to reduce the assumptions two parties make about each other when they exchange information” (Hohpe & Woolf, 2003).

Loosely-coupling often involves specific design patterns, i.e., solution schemes

for common problems that can be adopted to the specific situation. Many different kinds of design patterns exist. *Nota bene*, there are also integration patterns, as, for example, the ones proposed in the collection by Hohpe and Woolf (2003).

Following the idea of Hohpe's and Woolf's (2003) integration patterns (but restructured and extended), interoperability can be seen to be achievable on three different **architectural layers** (Wild & Sobernig, 2006): on the **presentation** layer through the inclusion of the direct user-interfaces, by **remoting** in the form of shared functions, or as **data integration** and dissemination through the replication, retrieval, filtering, etc. of data stemming from other systems (see Figure 1).

Furthermore, blog interoperability can be achieved in two different **interaction styles: push and pull**. In the push case modifications are propagated by the application managing the changed blog. It sends a message with information on the modifications to the servers storing the subscribed blogs. In the pull case the blogging application checks regularly if there was a change in the subscribed blogs and filters for updated data – thus generating a larger amount of overhead data.

Pulling requires two steps of communication, while pushing maintains state information and sends data only when necessary: it preserves information about clients' interests and pushes only relevant information. Pull interactions require many queries without effect, as weblogs usually do not change very rapidly (but at different times). As a consequence, pulling causes a larger communicative network load, especially when there is a larger number of clients. In the push approach, action is performed only when needed. The extensive performance analysis of push versus pull is described by Deolasee et al. (2001): they were able to show that for small temporal coherency requirements, pulling bears performance disadvantages.

Regarding **authentication and authorization**, the following statements can be derived from the scenario descriptions as outlined above and in accordance with other claims for the social software landscape in general (Downes, 2005; Bhatti et al., 2007; Wikipedia, 2007).

First, within an open learning situation, there is no need for explicit authentication, as the idea of public and distributed provision incorporates certain elements of anonymity and pseudonymity. Identity-resolution only takes place in the mind of the subscribers. There is no need and often no possibility for explicitly authenticating identity beyond disguised (or simply: 'not familiar') recognised or derived identities: no need to authenticate strangers. An external artefact, e.g., a blog feed, stands as *pars pro toto* for its creator(s).

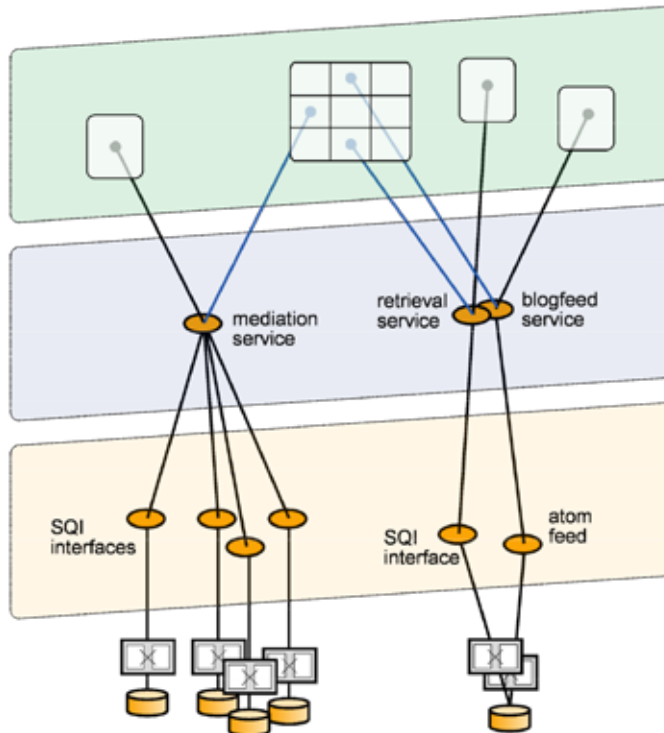


Figure 1. Architectural Layers (cf. Wild & Sobernig, 2006).

Second and furthermore, the idea of publicly providing feeds already turns around authorisation mechanisms of traditional access control mechanisms: a publicly available feed is syndicated, i.e., actively subscribed via pull or push-upon-request mechanisms. There is no need to have distributed authorisation mechanisms that would grant system-external subjects access rights to system-internal objects. Access is only granted in consent and under the control of a system-internal, thus mediating subject.

For these settings, capability-based credentials as such (when key- or ticket-bound, not name-bound) regulate access control better and can replace identity-based credentials of centralised authentication and authorisation protocols (cf. Bhatti et al., 2007; Miller et al., 2003; Levy, 1984).

4. Related Work

There are many attempts to bring interoperability to the blogosphere. These efforts are mainly targeted towards two levels of the interoperability stack: (meta-) data schemes and item collection exchange protocols between different blogging tools. The first kind came to birth with the RSS feeds formats and standards, the second relates to several emerging blogging application programming interfaces (APIs).

In the following section, these standards will be investigated. Furthermore, room will be given to an analysis of the application landscape available today and, subsequently, to a summary on these preceding efforts which offer starting points for an interoperability-architecture for distributed blog systems necessary to support collaborative learning settings.

4.1. Feed Standards

Meta-data standards define (XML) languages for describing collections of web content items and the information related to them. Collections are represented as feeds (also called ‘channels’). Content items are usually weblog posts, but also images, audio, video, or any resource published in the web. The origins of these formats date back as early as 1995 (Wittenbrink, 2005), although today only a few of them are still in use.

Although there had been various proposals for meta-data standards for news synchronisation – Wittenbrink (2005) counts eleven different standards in altogether 30 different versions –, today there are three main meta-data standards for content syndication: RSS 1.0, RSS 2.0, and Atom.

RSS 1.0 was developed later than the precursors of RSS 2.0 (that might be a bit confusing) and bases on a different technology: it deploys the resource description framework (RDF) of the W3C (Herman et al., 2007) and is not a simple XML derivative. It’s more difficult to use, however, to the benefits of its expressiveness (Wittenbrink, 2005).

The intellectual property rights of the second one, **RSS 2.0**, were donated to Harvard University (Winer, 2003) and its further development was frozen in 2002. It seems to be more popular, although it is less formally grounded and although the development process has been widely criticized. Some of the drawbacks from the interoperability point of view are that RSS 2.0 is not in an XML namespace and its content model does not permit well-formed XML mark-up. Maybe it is just ‘too simple syndication’.

The **Atom Syndication Format** (Nottingham & Sayre, 2007) is an XML-based

document format, developed by the Atom Publishing Format and Protocol (atompub) IETF working group (Winer, 2003). It is the result of a standardisation effort, being robust and complete. It was designed as to be most possibly flexible which is accomplished through extensions. An extension that covers threading has been proposed (Snell, 2006), which seems to be appreciated as a „robust specification for how to syndicate comments” (Reese, 2006).

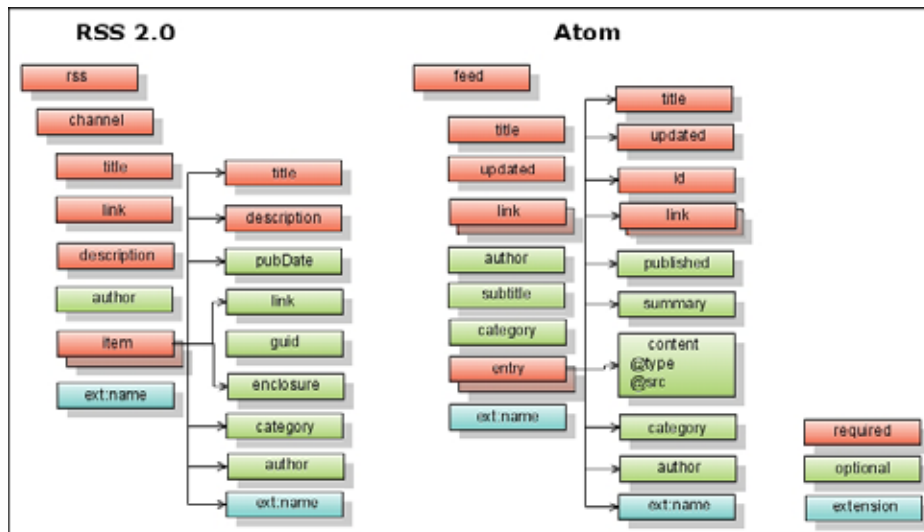


Figure 2. Core features of RSS 2.0 and ATOM compared.

There is no short answer on which of these three alternatives to choose. All of them share a common core (see also Figure 2). Depending on what is to be achieved, however, there are use-case specific advantages of the one or other format. RSS 1.0 is semantic-web technology and achieves more expressiveness through the separation of document format and content such that it allows for subsequent reasoning in the available (meta-) data. RSS 2.0 offers more simplicity in the specification (through fuzziness) and some advantages for provision of multimedia data, e.g., for podcasting. Atom’s development process is more open; the standard itself is also formally and completely defined. Transformations between these formats can be easily executed, although the interpretation of the transmitted data might then be more dependent on the interpreting applications.

Besides mark-up languages for feeds and items (including possibilities to annotate the origins of items), a meta-data format for exporting lists of feeds has been proposed by Winer (Winer, 2000): the **Outline Processing Markup Language (OPML)**. With these outlines, a list of feeds can be exchanged in a structured format, so-called blogrolls (Kalz et al., 2007) can be exchanged in a machine-readable format.

4.2. Blog Interaction Standards

Several specifications for specific kinds of **blog interaction** have been brought forward in the recent years.

The first xml-rpc specification overall and of course the first xml-rpc specification for publishing blog items directly was developed in 1998 for UserLand Frontier, the kernel behind the two UserLand products Manila and Radio (Winer, 1999). Subsequently, Winer was involved in developing another xml-rpc plus soap interface for Manila (Radke, Simmons, and Winer, 2000).

The first well-known API for publishing was developed by LiveJournal (around 1999), although it still seems to be used merely for Sixapart's LiveJournal. It passed simple key-value pairs for every request and response over HTTP (Livejournal, 2007). The next big API was developed for Blogger, also based on xml-rpc, the metadata for entries cannot be extended (Williams, 2001). In response to that problem, UserLand created the MetaWeblog API (Winer, 2002). Both, the blogger API and the MetaWeblog API are widely used. In 2007, Blogger itself replaced the blogger API by the generic Google Data API. Sixapart's Moveable Type adds further extensions on top of the blogger and MetaWeblog API: most notably this is the trackback extension invented by Ben and Mena Trott in 2002 (Reese, 2006). Starting as EchoAPI in 2003, the Atom Publishing Protocol emerged as an attempt to unify the various APIs and standardise it through the IETF. The AtomPP specification is still being developed today (Gregorio & de hOra, 2007). Although quite stable, there still might be changes in the future.

For brevity reasons, the following analysis concentrates on the most used APIs. In principle, the core of the **LiveJournal API** provides methods for authentication, publishing and updating postings (here called 'events'), downloading recent or distinct postings, listing the number of new postings and the number of postings per day, as well as some basic functionalities for listing other LiveJournal users as friends (LiveJournal, 2007). The API can be used via xml-rpc or plain http. There are three authentication methods: using plaintext passwords, using cookies (to prevent execution of actions unwanted by the user), and a token-based authentication that requests a so-called 'challenge' and subsequently combines it with the md5-encoded password to be (again) an md5-encoded authentication token.

Version 1.0 of the **Blogger API** simply provided methods how to create and edit a posting, to set and get templates how to display the main and the archive index pages, and how to get simply user information and blog memberships (Williams, 2001). This first version of the API was rather restricted regarding the meta-data capabilities: according to Doval (2003a, 2003b) the main drawback is that creation and editing of postings are completely restricted to the content of

the posting only. With the subsequent version based on Google's Data API, the xml-rpc calls were replaced by pure xml over http (Google, 2007). Its feature set still mainly centers around creation, maintenance, and reading of blog postings (and retrieving a list of blogs). However, methods to create, retrieve, and delete comments have been added.

Authentication can be performed via username/password login and subsequent token-based authorisation. Additionally, there is a mechanism (called 'AuthSub proxy authentication') allowing to delegate authentication from an external webapplication, so that a user can login via the Google authentication service to pass through the authorisation token to a webapplication through calling an exit url with appending the authorisation token.

The **MetaWeblog API** was the first interaction protocol that used a feed format specification to publish information onto a server. The „MetaWeblog API uses an XML-RPC struct to represent a weblog post. Rather than invent a new vocabulary for the metadata of a weblog post, we use the vocabulary for an item in RSS 2.0. So you can refer to a post's title, link and description; or its author, comments, enclosure, guid, etc. using the already-familiar names given to those elements in RSS 2.0" (Winer, 2002). Elements from different namespaces can also be mixed into the creation and update activities.

This API solved many problems, but the solution is quite complicated. Some of the problems remained the same due to xml-rpc specification, like implementation-dependent time zones and mismatch in date formats. Additionally, the MetaWeblog API provides methods to add media objects (binary uploads), getting a list of categories, and for fetching recent entries (Winer, 2002). Authentication is provided by transmitting passwords in the clear.

As a result of the shortcomings of commercially developed APIs, a working group in the Internet Engineering Task Force (IETF) was established that works on the standardisation of a the **Atom Publishing Protocol** (also short 'atomPP'). More extensive background information on why atomPP was established is published in Pilgrim (2003). The protocol along with the feed standard (Atom Syndication Format) was created by IETF's Atom Publishing Format and Protocol Working Group (known simply as 'atompub') for use not only with blogs but also other Web resources (Hoffman & Bray, 2006; Pilgrim, 2003; Snell, 2006). It defines a standard and interoperable way to create and edit on-line resources.

„The Atom Publishing Protocol is an HTTP-based approach for creating and editing Web resources. It is designed fundamentally around the idea of using the basic operations provided by the HTTP protocol (such as GET, PUT, and DELETE) to pass around instances of Atom 1.0 Feed and Entry documents that represent things like blog entries, podcasts, wiki pages, calendar entries and

so on” (Snell, 2007).

The protocol is based on HTTP, and follows REST architectural principles. The http ‘verbs’ mentioned in the text by Snell (2006) above, refer to using atom 1.0 feeds and entry documents:

- POST to add new entries and media files to a collection
- GET to retrieve a service document, collections, entries or media files
- PUT to update entries and media files
- DELETE to delete entries and media files (identified by their item url)

AtomPP today is in its 15th draft version. The protocol specification is not finalised yet, but the protocol core seems to be rather stable. Currently, there are several implementations of AtomPP (Intertwingly, 2007) such as IBM Lotus Connections or Flock.

The atomPP defines the following concepts for a publishing server. A ‘service document’ is an xml-document that describes collections on the server and accepted mime-types. ‘Collections’ are sets of items (atom entries or media files). Collections are represented by atom feeds. ‘Entries’ are items in a collection, as described in atom feed standard. AtomPP defines a new type of link for the atom entry, i.e., rel=“edit”, that is used to modify the entry. By using the basic http verbs, entries in collections can be retrieved, updated, stored, and removed. Through the service documents, generic directory listings can be retrieved allowing for discovery of collections.

Transmitting data between different systems can be realised by transforming local data schemes to atom and communicating them or about them via the atom publishing protocol through http (see Figure 3).

The plans of the atompublish charter even go further and include support for multiple authors, user management (including the setting of preferences), and getting or setting of related resources such as comments or templates (Hoffman & Bray, 2006).

As the atomPP is a REST-style (Fielding, 2000) http protocol, it is very consistent with the whole web architecture: it is well-defined data model, uses doc-literal style web services not rpc, deploys xml with namespaces, is bound to http, and is secure (Snell, 2006).

Several so-called ‘linkback’ extensions in addition to the publishing APIs have been proposed. Most prominent among these extensions are trackbacks and pingbacks. **Trackbacks** are a mechanism to inform the blog application A of an

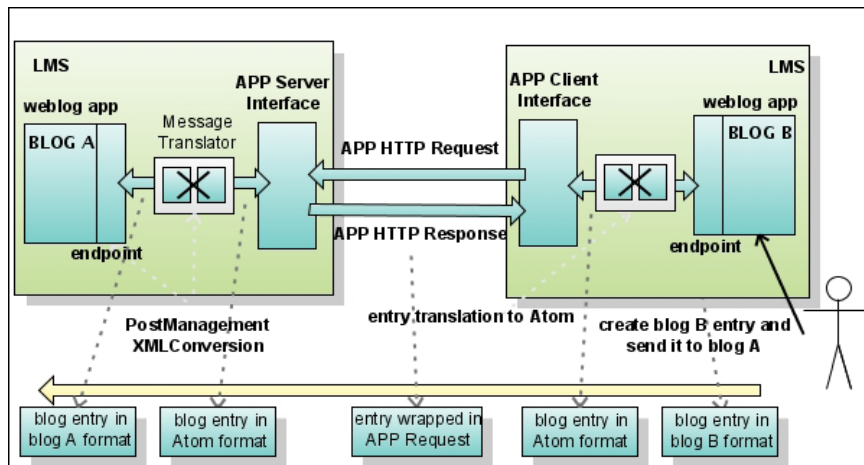


Figure 3. APP Architecture Visualization.

author x of a posting that is being referred to in the blog B of an author y in via a ping: instead of using the comment function in blog A, the author y can reply in his own blog to an external posting. Subsequently, her blog application B will inform A with a short REST-style remote procedure call through http (Brahms, 2007; Trott & Trott, 2002).

A similar, but less spam-vulnerable mechanism for informing about citations and replies has been proposed by Langridge & Hickson (2002) – the **pingback** specification. The procedure follows the same principle, i.e., an xml-rpc request is executed from the citing system to the system with the original posting, the page with the linking posting is retrieved and parsed for the citation information, thus making it less vulnerable to spam.

Both the pingback as well as the trackback mechanism have been re-used (not to say: ab-used) to support indexing of blog postings in aggregator services on the web (for an overview on pingback-based indexing services see Pchere (2005), see Paquet & Pearson (2004) for an example on topic-oriented trackback-based indexing services).

As can be seen from the **API comparison** in Table 1, the most extensive protocol is the atom publishing protocol that realises almost all identified capabilities needed for publishing and retrieving collections and entries, competing hard with Blogger's new GData API. At the same time, the comparison table shows clear shortcomings: even simple management functionalities that support indexing and distribution of blogs are only supported via external specifications such as pingbacks and trackbacks.

Feature / API specification	Publish Posting (put, edit, del)	Publish Media Objects	Get Categories	List Postings	Get Posting	Publish Comment	Retrieve Comment	Instant Linkbacks	List 'friends' on same server	Get Blogrolls	Advertise Feeds	Request Update Pings	Instant Indexing (Update Ping)	Extensive Content Model	Simple Content Model
LiveJournal	x			x	x				x						x
Blogger API 1.0	x									x					x
Blogger GData API	x			x	x	x	x			x				x	
MetaWeblog API	x	x	x	x	x									x	
Atom Publishing Protocol	x	x	x	x	x	+	+								x
Trackback Specification						x		x					o		x
Pingback Specification						x	x	x					o	x	

o = partially through unintended use; x = fully; + = optionally possible

Table 1. Overview on Differences between the APIs.

Networking interactions seem to be the major shortcoming of today's APIs. Passive networking interactions via these APIs are only supported by early APIs and the GData API – and even there only to a limited amount (either listing friends or getting blogrolls of users). There is no support at all for active networking interactions, which are essential for articulation work in collaborative learning processes: only the abuse (i.e., unintended use of the specification in platforms like technorati) of trackbacks and pingbacks allows for instant indexing, proactive feed management (advertising feeds, requesting subscriptions) between users is overall not supported. The core publishing process is fully supported by several of the state-of-the-art APIs, whereas comment management in distributed settings can only be achieved by deploying trackbacks and pingbacks in addition to existing APIs.

4.3. Existing Blogging Tools

The number of **blog applications** is huge and still growing. Many of them are provided as free hosting services (like wordpress.com). The collection of the E-Learning Centre (2006) lists popular tools for creating blogs, mobile blogs, audio blogs (a.k.a. podcasts), video blogs (a.k.a. vblogs), and aggregators to read

blog feeds. Apart from quite heterogeneous content representation, the majority provides feeds for content syndication in standard formats – RSS 2.0 or Atom. Some of them supply also an API allowing to create new blog posts, edit or delete existing posts, query for posts that match particular criteria, some of them even perform some user management actions. Migration is often supported. WordPress, for example, currently supports the import of data from a variety of other content publishing platforms and from any RSS feed (Wordpress, 2007). Several popular blogging systems are: drupal, elgg, manila, MovableType, LiveJournal, WordPress / WordPress MultiUser, b2evolution, textpattern, nucleus, roller, pivot, and pLog (today re-branded to ‘LifeType’) (Farmer, 2005; Bauer, 2004). All of these, except Manila and Moveable Type, are open-source applications.

Standard / System	RSS 1.0	RSS 2.0	ATOM	LiveJournal	Blogger API 1.0	Blogger GData API	MetaWeblog API	AtomPP	Trackback	Pingback	Open Source
Drupal		x	x				x		x	x	x
Elgg		x					x				x
Manila		x	x		x		x		x		
Movable Type	x	x	x					x	x		
LiveJournal		x	x	x	x			x			x
WordPress ^H	x	x	x		x		x	x	x	x	x
b2evolution		x	x		x				x	x	x
Textpattern		x	x		x		x				x
nucleus	x	x	x		x		x		x		x
roller		x	x		x		x	x	x	x	x
pivot		x	x				x		x		x
pLog / LifeType	x	x	x		x		x	x	x		x

Table 2. Current Standard Support Overview.

As can be seen from the overview in Table 2, standard support among the blog systems is quite heterogeneous. All systems support RSS 2.0, closely followed by atom, whereas RSS 1.0 feeds are only provided by a limited number of systems. Among the APIs, the blogger 1.0 and the MetaWeblog API compete closely; support for the Atom Publishing Protocol, however, is growing. Trackbacks are supported by many systems, pingbacks are already offered by several ones.

A **blog aggregator** or a feed reader is the common solution to consuming syn-

icated content. It can monitor many sites and sources providing near real-time updates to one location. There are a number of different types of readers: web-based, desktop, based on specific application like Microsoft Outlook or Mozilla Thunderbird. For here, we will focus on web-based feed readers.

Aggregators can be classified into personal and community aggregators. Among personal aggregators, we can find web-based tools and desktop tools. Personal aggregators are intended for personal use. A user subscribes to her feeds of interest and manages them in a way similar to a mail client. Analogously to mail clients, the application can either be installed on a web server (then accessed through a browser) or can be provided as a desktop application, independent or integrated with a browser. Desktop aggregators are, for example, Personal Learning Environment (PLEX) and the Firefox derivate Flock.

Standard / System	RSS 1.0	RSS 2.0	ATOM	LiveJournal	Blogger API 1.0	Blogger GData API	MetaWeblog API	AtomPP	Trackback	Pingback	Open Source
Amphetadesk	x	x									x
Feed on Feeds	x	x	x								x
Html / Flash RSS Reader	x	o									?
Gregarius	x	x	x								x
HEP Message Server		x	x				x				x
Lilina	x	x	x								x
Lyline	x	x	x								x
MyHeadlines	x	x	x		x						x
News-Maniac		x	o								x
Newspipe	x	x	x								x
phpNewsfork	o	o									x
Planet Planet	x	x	x								x
Python Desktop Server		x			x		x	x			x
Railsplanet	x	x	x					x			x
Rawdog	x	x	x								x
reBlog	x	x	x				x	x	o		x
Reptile		?									x
Rippy The Aggregator		x	o								x
Rnews	x	x	x								x
RSS Feed Magic	x	x									o
RSS Merge	x	x									x
Spycyroll / pyBlagg	x	o									x
TALAggregator		x									o
Tiny Tiny RSS	x	x	x								x
Urhcin	x	x									x

Table 3. Overview on Standard Conformance of Aggregators.

Community aggregators gather web feeds relevant to a group of interest using a stable feed list. They aggregate, for example, all feeds from a course. They merge all items in the feeds, subsequently provide one merged feed, and display all items additionally on a web page.

News on Feeds (2007) lists 13 different web-based aggregators. Additionally, Hebig (2005) lists another 13 aggregators. No longer existing aggregators were excluded from the analysis, two more aggregators were added.

As can be seen from the overview in Table 3, many of the aggregators systems support all important content standards (12/25 support all). Regarding APIs, however, the overview shows clearly that almost no system supports an API. This is even more astonishing as the APIs contain several synchronisation routines and abuses of trackback and pingback are very popular with commercial online services such as technorati to realise indexing updates. Few systems contained proprietary APIs, that can be used to synchronise the online aggregator with a desktop application. Regarding subscription management and feed advertisement, none of the system provides any support.

Let's additionally consider three popular learning environments widely used in education: Moodle, SAKAI, and .LRN. Additionally, we compare these systems with IVA, one of the smaller open-source software projects in technology-enhanced learning.

In **Moodle**, the modular object-oriented dynamic learning environment, blogs are user based - each user has their own blog. Users can also create tags that can be associated with blog entries - admins can create site level tags, teachers can create course level tags, and students can create their own list of tags. Administrators can restrict the visibility of users' blog entries to themselves, to a group / course level, or to the world.

In **SAKAI** the user creates a team-oriented blog which means that all published information can be revised, modified and appended to by any member of the particular team. All team members can also add comments to any blog entry¹. Post fields include title, shortText, date, state, creator, keywords, elements, authors and comments.

.LRN is a learning system built upon the Open Architecture Community System (openACS). .LRN is actively developed by the Vienna University of Economics and Business Administration, which also delegates a member of the board of directors.

IVA (<http://www.htk.tlu.ee/iva>) is a web-based learning management system, which has been developed by Tallinn University. It is built on Zope, an open-source content management system written in Python.

¹ <http://bugs.sakaiproject.org/confluence/display/BLOG/Home> (accessed Jan, 2007)

Standard / System	RSS 1.0	RSS 2.0	ATOM	LiveJournal	Blogger API 1.0	Blogger GData API	MetaWeblog API	AtomPP	Trackback	Pingback	Open Source
Moodle		×									×
Sakai	×	×									×
.LRN	◦	×	◦		×	×		×	×	×	×
IVA		×								◦	×

Table 4. Overview on Blog Tool Support in LMSs.

As can be seen from the overview in Table 4, standard support among educational applications is even worse, with the respectable exception of .LRN which's openACS weblogger module supports even several APIs plus trackbacks and pingbacks.

4.4. Current Shortcomings

As we have shown through our feed standard analysis above, no clear recommendation can be made in favour of one of the three major feed data standards – RSS 1.0, RSS 2.0, or Atom. All of them have different advantages, all of them are extensible, and all of them can – eventually with interpretation disadvantages on the application sides – be transformed into each other. As other iCamp activities already started off with RSS 2.0, we recommend to stick to that decision (until better reasons point us to a different standard).

By investigating publishing APIs and their enhancements that have been brought forward in recent years, we come to the following conclusion: Standard interfaces exist that support the core publishing process (as necessary for remote controlling a blog with a desktop editor), support for networking interactions is virtually not available. Trackback and pingback, alas, have made the first attempts into that direction, although they were not originally designed for it. We conclude that this major shortcoming in the area of active networking needs to be solved in order to comply with our requirements which we sketched in the scenario section of this contribution.

Looking at state-of-the-art blog applications, we see that many of them support an extensive set of both API and feed data standards, however, naturally reflecting their lack of networking support. Aggregators usually do not support any API standard, which – in case of pingbacks and trackbacks – is especially astonishing. Most of them even lack in support of content standards which prevents them from fulfilling the job they were designed for. Again and even

more astonishing, aggregators lack capabilities for subscription management and feed advertisement. Most learning environments developed own blogging components. Their standard support is even worse with the notable exception of .LRN that – w.r.t. standards compliance – competes in the same class as the best stand-alone blogging tools

As our three analysis sections show us, the major shortcomings lie in realising interoperability for feed management.

5. The Missing Link: The FeedBack Specification

The process of collaborating via blogs can be divided into two independent sub-steps, i.e., the management of feeds and communication channels including authorisation (the articulation work) and the exchange of items or item collections (the materialisation, the content transmission itself). The following section describes the missing link, a specification for managing feed subscriptions in a distributed setting which complements existing standards as analysed in Section 3.

Therefore, we describe data exchange, interaction steps, and state-transitions between two non-identical blog systems and their users, when communicating management information about particular feeds and specific items in these feeds. This model realises an interface specification for user-centred distribution of feed aggregation activities which is both prerequisite and basic infrastructure for blog-based collaboration.

Aggregation services are already ‘abusing’ the pingback specification in so far as they are using pingback xml-rpc calls to inform about new and updated items and no longer inform about replies to existing blog postings. However, at the same time, there are no standardised options to inform a system about the existence of a feed and about updates which would enable more efficient management in synchronisation.

This document proposes a set of xml-rpc calls to transport blog management information from one system to another. It is light-weight in so far, as implementation is made as easy as possible and dependencies on other components are reduced to a minimum. The whole communication process apes human behaviour and shifts control to the user wherever possible.

5.1. Communication Process

There are three important steps when managing feeds. First, the communication partner has to be informed about the opportunity of subscribing to a blog. Second, this communication partner has actively to acknowledge whether she wants to be informed about future updates of the advertised blog, i.e. to confirm the subscription. Finally, the communication partner has to be pinged about every update that is available in order to allow for efficient synchronisation. To specify the involved xml-rpc calls, relevant parameters, and feed data to be exchanged, we are going to illustrate the communication process through the construction of an example between the two fictive persons ‘Steinn’ and ‘Fridolin’ as well as their blogging systems ‘Moodle’ and ‘Wordpress’.

5.1.1. Advertise Subscription Opportunity = < feedback.offer >

Fridolin reads Steinn’s blog feed at

`http:// steinn.at/rss.php`

and decides that Steinn should subscribe to his blog. So he has his Moodle tell Steinn’s Wordpress (which is able to aggregate feeds) to subscribe to a feed. This subscription request usually then has first to be acknowledged by the owner of the blog who was offering the

X-Feedback: `http://steinn.at/xmlrpc.php`

or the

`<link rel="feedback" href="http://steinn.at/xmlrpc.php">`

in his blog. In this case, the feed opportunity is offered to Steinn and Steinn’s url for the xml-rpc calls is ‘`http://stein.at/xmlrpc.php`’. When contacting this ‘feedback.offer’ xml-rpc endpoint, the call should contain the url of the requesting system’s full-length blog feed and the url of the blog feed for which the subscription request was meant.

```
xmlrpc = xmlrpc.server("http://steinn.at/xmlrpc.php")
xmlrpc.feedback.offer (
    "http://fridolin.at/rss.php",
    "http://steinn.at/rss.php"
)
```

Proposal: It might be interesting to replace this ‘target’-feed address alternatively with some kind of ‘ID in URL format’.

5.1.2. Acknowledgement of Subscription = < feedback.request >

When Steinn acknowledges to subscribe, the requesting system (Fridolin) is

informed about this by fetching the feed, parsing for the 'feedback' xml-rpc url and by executing a 'feedback.request' xml-rpc call. The requested system will from now on try to send update notifications to inform about changes. In detail, this means to:

```
fetch( "http://fridolin.at/rss.php" )
```

and store all entries in the local database, then parse the http-header for

```
X-feedback: http://fridolin.at/xmlrpc.php
```

In case there is no line in the http-header, the feed must be parsed for a matching link-rel-tag:

```
<link rel="feedback" href="http:// fridolin.at/xmlrpc.php">
```

Then Steinn's system is creating a unique token to be shown whenever Fridolin's blog is sending update information in the future, so that Fridolin's system can be identified easily and authorizes itself based on this token (so to say the 'one-purpose passport' for a system): Steinn's token for Fridolin's system is 'zZ12H33ux4jh9703D'. Next, Steinn's system performs an xml-rpc call to Fridolin's 'feedback.request' url and transmits the url of the desired blogfeed, it's own 'feedback.notify' xml-rpc address, and the token:

```
xmlrpc = xmlrpc.server("http://fridolin.at/xmlrpc.php")
xmlrpc.feedback.request(
    "http://fridolin.at/rss.php",
    "http://steinn.at/xmlrpc.php",
    "zZ12H33ux4jh9703D"
)
```

Fridolin's system stores this token to use it in the notification calls of 'feedback.notify' to 'http://stein.at/xmlrpc.php' whenever an update is available.

If there is not 'feedback' xmlrpc url or the remote procedure call 'feedback.request' fails, the requesting system has to care for regularly fetching the feed itself.

5.1.3. Inform about Updates = < feedback.notify >

Now Fridolin writes a new entry, update or deletes an existing one, and wants to inform Steinn's blog to fetch the changes:

```
xmlrpc = xmlrpc.server("http://steinn.at/xmlrpc.php")
xmlrpc.feedback.notify(
    "http://fridolin.at/rss.php?p=55", ### or, e.g., blabla.at/?p=55
    "zZ12H33ux4jh9703D" ### re-use the token from STEP 2
)
```

Steinn's system knows that it is Fridolin because of the unique token. Steinn's system should parse the RSS and check whether the `<channel><link /></channel>` is the same as the one, it originally subscribed to (in this case it needs to be "http://fridolin.at/rss.php"). Then it parses the content of the RSS feed and indexes, updates, or inactivates the contained article(s).

When Steinn decides that he no longer wants to get update information from Fridolin's blog, he just needs to make his system reject Fridolin's next 'feedback.notify' call with an error message '301'.

The address "http://fridolin.at/rss.php?p=55" MUST NOT be deleted after it has been fetched by Steinn's system as long as the article(s) = item(s) exist(s) in Fridolin's blog system. It SHOULD BE a link to the e.g. article or articles.

Proposal: alternatively to the token, a username plus password combination for basic-auth and succeeding atom-PP synchronisation could be passed.

5.2. Validity Check: Reference Implementation

To demonstrate the applicability of the missing link 'FeedBack' that complements existing standards and specifications in the blogosphere to also support articulation work for managing offer, subscription, and update routines in collaborative learning processes, the following section describes one of the scenarios depicted in Section 2 in more detail. This scenario will be the simple feed management scenario, because it is easy to follow and because other scenarios are merely more complex wirings of these simple management processes.

To realise the simple feed management scenario, usually at first, a sample request has to be sent to a FeedBack enabled blog – resulting into a pending offer such as depicted in Figure 4 in the shaded box to the left. The subscription offer can be accepted or rejected resulting in a one-time rejection, so future offers of the same feed will not be turned down. It is thinkable to add a 'auto reject' option that automatically turns down offers of a rejected feed (to avoid offer spamming).

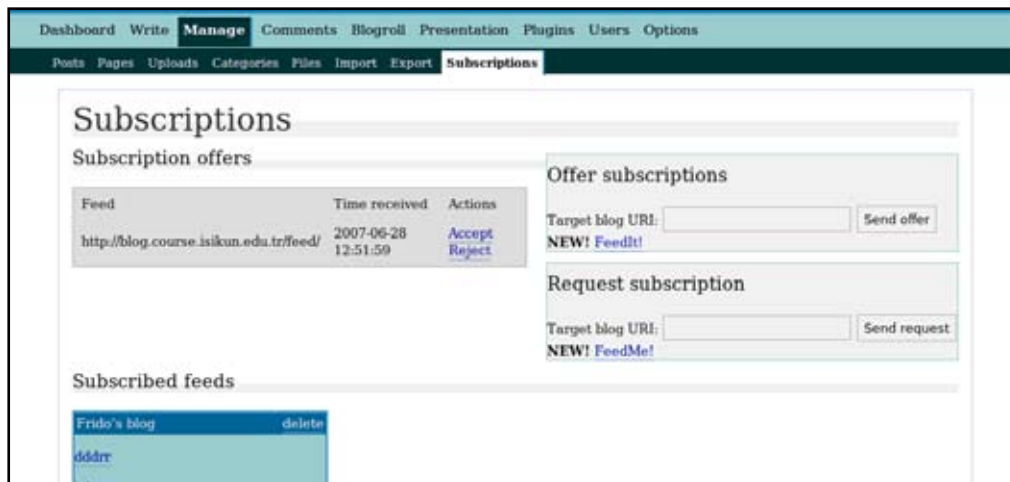


Figure 4. Pending Offer.

Once this offer has been accepted and the request updates remote procedure call was successfully dispatched to the remote system, the newly subscribed feed is displayed in the aggregator panel at the lower bottom of the screen, as depicted in Figure 5: all items of subscribed feeds (pushed-subscription!) are listed here and can be conveniently accessed. We plan to separate between the management (offer, accept, request) side from the aggregator side, so that there are different user interfaces for these functionalities.

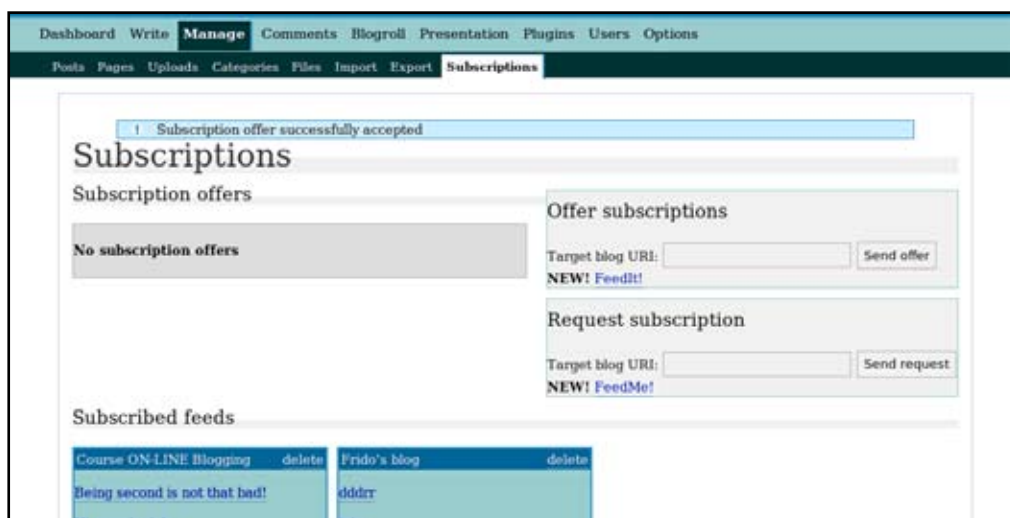


Figure 5. Accepted Offer.

From now on, the background update processes of the remote system will ensure that update notifications will be transmitted whenever there are changes in the subscribed feed. Update notifications should be cancelled implicitly by denying an update notification request.

To make feed management even easier, two droplets have been designed: the first one (see Figure 6) can be activated on every external blog application (that supports FeedBack) to immediately add the feed of the blog to the list of subscriptions.

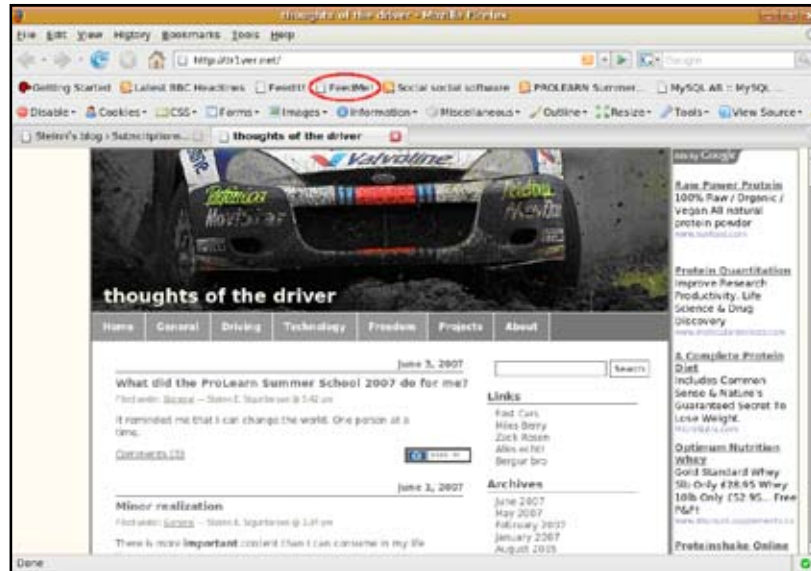


Figure 6. Droplet 'FeedMe'.

When executed, the feed is fetched, parsed for the rpc-service address, the subscription request is sent, and – when successful – future updates will be received for the aggregated list of feeds (see Figure 7).

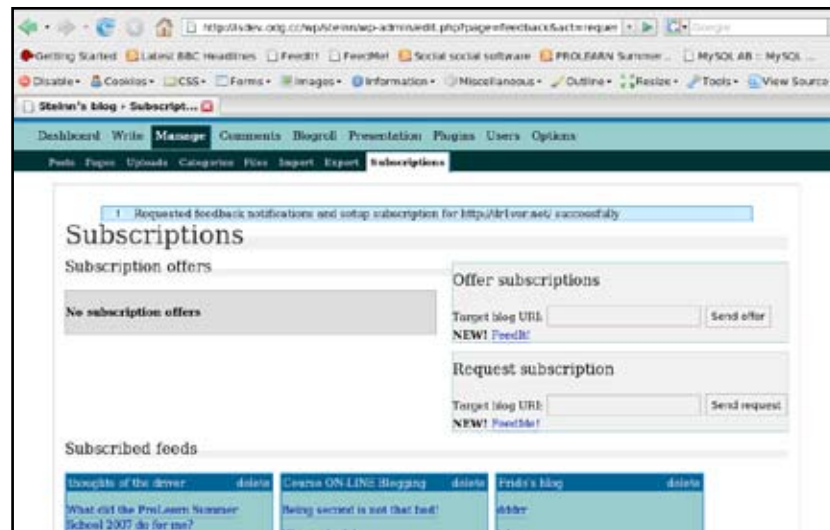


Figure 7. Successful droplet activity of 'FeedMe'.

The second one, the so-called 'FeedIt' droplet, turns this process around: A feed offer can be conveniently sent to any FeedBack-compliant external blog, resulting in the execution of the xml-rpc offer service by your own blog application.

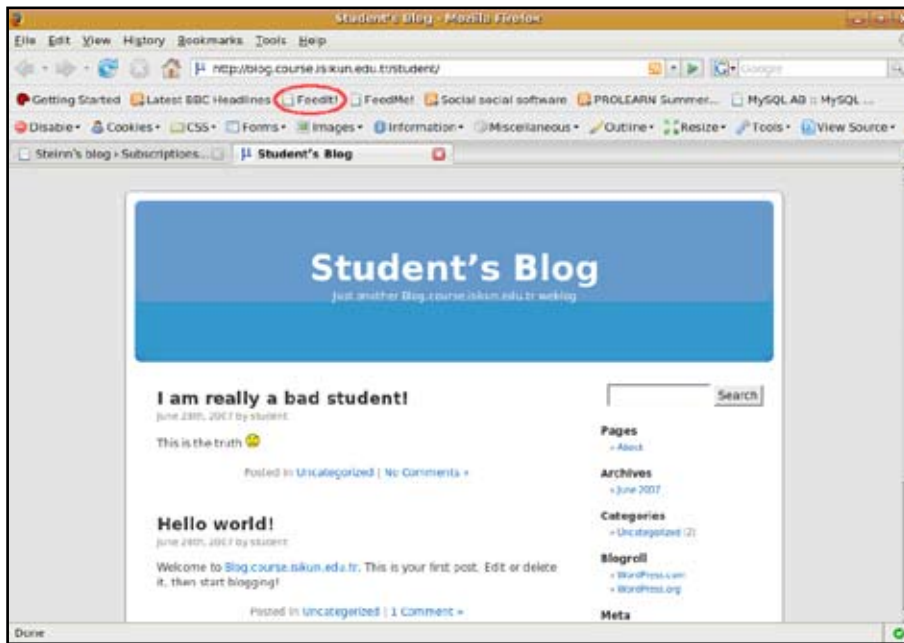


Figure 8. Droplet 'FeedIt'.

Figure 9 shows the status message of this offer remote procedure call as displayed by your own blog application.

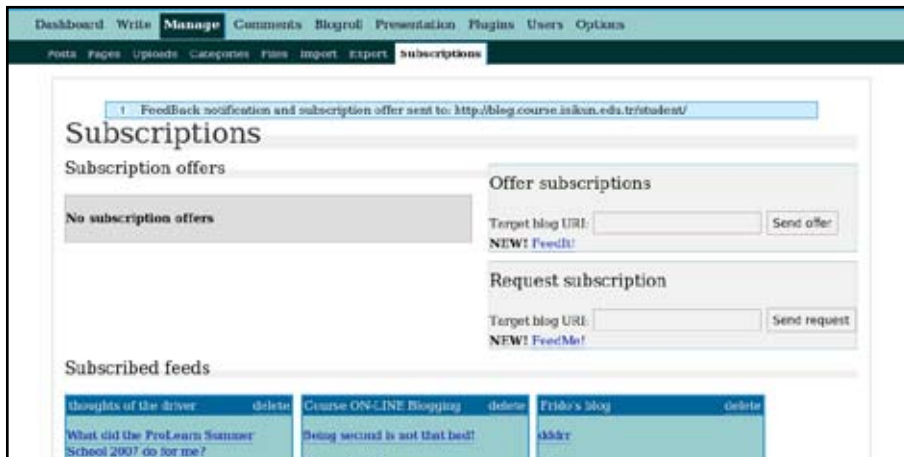


Figure 9. Successful Droplet Activity: Offer was sent.

Figure 10 depicts the status message after this incoming offer has been accepted in the remote blog application to which the offer has been sent.

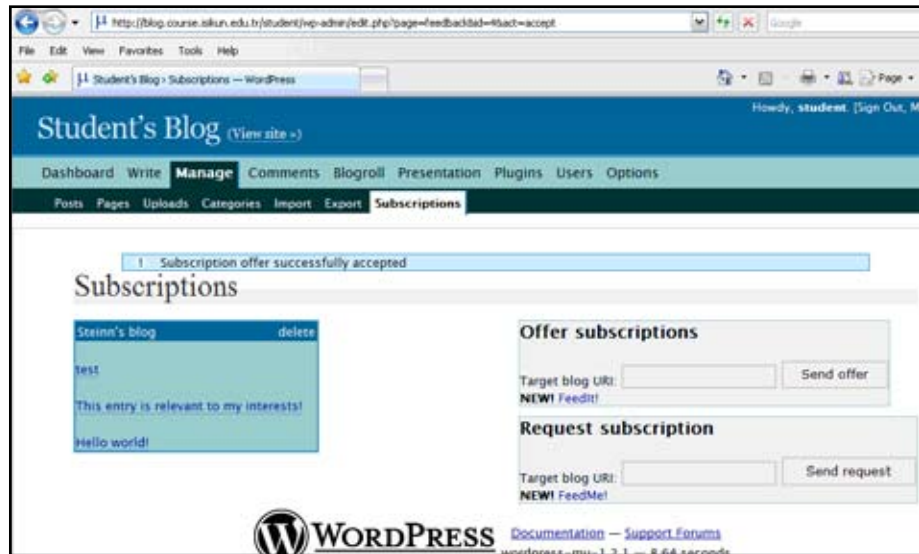


Figure 10. The other side: offer accepted in the student's blog.

6. Discussion & Outlook

Within this contribution we have extensively analysed the status quo of the technology behind the blogosphere, have identified a major shortcoming from the perspective of its applicability in collaborative learning situations, and have proposed a solution to solve it: FeedBack complements existing standards with a specification on how to support management processes for advertising of, subscription of, and update notifications on existing blog feeds. We have prototypically tested this specification with implementations for a standard blogging system, WordPress and WordPress_µ, and a learning management system, IVA. Currently, we are working on additional implementations, a public release of the WordPress module is planned. The prototype of this emerging network shows us that it is feasible. Filling this space with life to turn it into a sustainable place sets yet another challenge for us which we plan to meet in the next trials.

7. References

- A. Shakya, H. Takeda, I. Ohmukai, and V. Wuwongse: A Publication Aggregation System Using Semantic Blogging, in G. Li, Y. Liang and M. Ronchetti eds., *The Semantic Web — ASWC 2006 Workshops Proceedings*, pp. 55–62, Jilin University Press (2006)
- Bakshi, K., Karger, D.R., Quan, D., (2004), *The Message without the Medium: Unifying Modern Messaging Paradigms through the Semantic Web*, <http://haystack.lcs.mit.edu/papers/www2004-messaging.pdf> (accessed Jan, 2007)
- Bauer, Elise (2004): *An Overview of the Weblog Tools Market*, http://www.elise.com/web/a/an_overview_of_the_weblog_tools_market.php, last access: July 5th, 2007
- Beale R. *Mobile Blogging: Experiences of Technologically Inspired Design*. Proceedings of ACM Conference on Human Factors in Computing Systems, CHI'06. Montréal, Canada. April 22-27, 2006.
- Bhatti, Rafae; Bertino, Elisa; Ghafoor, Arif (2007): *An Integrated Approach to Federated Identity and Privilege Management in Open Systems*, In: *Communications of the ACM* 50(2), ACM Press, pp. 81 – 87
- Brahm, Taiga (2007): *Blogs – technische Grundlagen und Einsatzsszenarien an Hochschulen*, In: Seufert, Brahm (Eds.): “Ne(x)t Generation Learning”, SCIL-Arbeitsbericht 12, SCIL, <http://www.scil.ch/publications/reports/2007-02-euler-seufert-next-generation-learning.pdf>, last access: June 27th, 2007
- Brahm, Taiga (2007): *Blogs – Technische Grundlagen und Einsatzsszenarien an Hochschulen*, In: Seufert, Brahm (2007): *Ne(x)t Generation Learning*, SCIL-Arbeitsbericht 12, SCIL, Switzerland, pp. 69 – 89
- Cayzer, S. *Semantic Blogging and Decentralized Knowledge Management*. *Communications of the ACM*, 47 (12), 48-52.
- Chin, Alvin; Chignell, Mark (2006): *A Social Hypertext Model for Finding Communities in Blogs*, In: *Proceedings of the Hypertext 2006 (HT'06)*, Odense, Denmark, pp. 11 – 22
- De Moor, Aldo; Efimova, Lilia (2004): *An Argumentation Analysis of Weblog Conversations*, In: In: Aakhus, Lind (eds.): *Proceedings of the 9th International Working Conference on the Language-Action Perspective on Communication Modelling (LAP 2004)*, New Brunswick, USA, pp. 197-211
- Deolasee, P., Katkar, A., Panchbudhe, A., Ramamritham, K., Shenoy P., (2001), *Adaptive Push-Pull: Disseminating Dynamic Web Data*, ACM 1-58113-348-0/01/0005, <http://www10.org/cdrom/papers/pdf/p269.pdf> (accessed Jan, 2007)

- Doval, Diego (2003a): A Review of Blogging APIs, <http://www.dynamicobjects.com/d2r/archives/001921.html>, last access: July 26th, 2007
- Doval, Diego (2003b): Blogging APIs: a mini How-To, <http://www.dynamicobjects.com/d2r/archives/001929.html>, last access: July 26th, 2007
- Downes, Stephen (2005): Authentication and Identification, In: *Instructional Technology & Distance Learning*, Vol. 2, No. 10, DonEl Learning Inc.
- eLearning Centre (2006): Products & Services: Blogging Tools, eLearning Centre, Learning Light, <http://www.e-learningcentre.co.uk/eclipse/vendors/weblogging.htm>, last access: July 5th, 2007
- Farmer, J. (2005), Centred Communication: Weblogs and aggregation in the organisation, Blogtalk Downunder, May 19-22 Sydney, http://incsub.org/blogtalk/?page_id=54 (accessed Jan, 2007)
- Fielding, R.T., (2000), Architectural Styles and the Design of Network-based Software Architectures, Representational State Transfer (REST), University of California, Dissertation, http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (accessed Jan, 2007)
- Google, Inc. (2007): Blogger Data API Developer's Guide: Protocol, http://code.google.com/apis/blogger/developers_guide_protocol.html, last access: June 25th, 2007
- Gregorio, J.C.; de hOra, B. (2007): The Atom Publishing Protocol, draft-ietf-atompub-protocol-15.txt, <http://ietfreport.isoc.org/all-ids/draft-ietf-atompub-protocol-15.txt>, IETF
- Gurzick, David; Lutters, Wayne G. (2006): From the Personal to the Profound: Understanding the Blog Life Cycle, In: *Proceedings of the CHI 2006*, Montreal, Canada, ACM Press
- Harri-Augstein, Sheila; Webb, Ian M. (1995): *Learning to Change*, McGraw-Hill, London
- Hebig, Heiko (2005): RSS Feed Reader / News Aggregators Directory, <http://hebig.org/blogs/archives/main/000877.php#scripts>, last access: July 27th, 2007
- Herman, Ivan; Swick, Ralph; Brickley, Dan (2007): Resource Description Framework (RDF), W3C, <http://www.w3.org/RDF/>, last access: July 5th, 2007
- Herring, Susan; Scheidt, Louis; Bonus, Sabrina; Wright, Elijah (2004): Bridging the Gap: A Genre Analysis of Weblogs, In: *Proceedings of the 37th Hawaii International Conference on System Sciences (HICSS-37)*, IEEE Computer Society Press
- Hoffman, Paul and Bray, Tim (2006): Atom Publishing Format and Protocol Charter (atompub), <http://www.ietf.org/html.charters/atompub-charter.html> (accessed Feb, 2007)

- Hohpe, G., Woolf, B., (2003), *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*, Addison-Wesley Professional
- Hohpe, Gregor and Woolf, Bobby (2003): *Integration Patterns Catalogue*, <http://www.enterpriseintegrationpatterns.com/eaipatterns.html> (accessed June 20, 2007)
- Intertwingly (2007): *Implementations of the Atom Publishing Protocol*, <http://intertwingly.net/wiki/pie/Implementations>, last access: July 5th, 2007
- Kalz, Marco; Specht, Marcus; Klamma, Ralf; Chatti, Mohammed; Koper, Rob (2007): *Kompetenzentwicklung in Lernnetzwerken für das lebenslange Lernen*, In: Dittler, Kindt, Schwarz (Eds.): *Online-Communities als Soziale Systeme*, Waxmann, New York/München/Berlin
- Karger, D.R., Quan, D., (2004), *What Would It Mean to Blog on the Semantic Web?*, *Journal of Web Semantics*, Volume 3 Issue 2, <http://theory.csail.mit.edu/~dquan/iswc2004-blog.pdf> (accessed Jan, 2007)
- Kosanke, K. (2005): *ISO Standards for Interoperability: a Comparison*. In: *Pre-Proceedings of the INTEROP-ESA 2005*, pp. 59–67, Geneva.
- Krause, Steven (2004): *When Blogging Goes Bad: A Cautionary Tale About Blogs, Email Lists, Discussion, and Interaction*, <http://english.ttu.edu/KAIROS/9.1/praxis/krause/index.html>, last access: July 5th, 2007
- Kumar, Ravi; Raghavan, Prabhakar; Novak, Jasmine; Tomkins, Andrew (2003): *On the Bursty Evolution of Blospace*, *Proceedings of the WWW2003*, Budapest, Hungary, ACM Press, pp. 568 – 576
- Langridge, Stuart; Hickson, Ian (2002): *Pingback 1.0*, <http://www.hixie.ch/specs/pingback/pingback>, last access: July 5th, 2007
- Levy, Henry (1984): *Capability-Based Computer Systems*, Digital Equipment Corporation
- Livejournal (2007): *Client/Server Protocol*, <http://www.livejournal.com/doc/server/ljp.csp.protocol.html>, last access: July 5th, 2007
- Miller, Mark; Yee, Ka-Ping; Shapiro, Jonathan (2003): *Capability Myths Demolished*, Technical Report SRL2003-02, Systems Research Laboratory, Johns Hopkins University
- Nardi, Bonnie; Schiano, Diane; Gumbrecht, Michelle; Swartz, Luke (2004): *Why We Blog*, In: *Communications of the ACM 47(12)*, ACM Press, pp. 41 – 46
- News on Feeds (2007): *News Aggregators*, <http://www.newsonfeeds.com/faq/aggregators>, last access: July 27th, 2007
- Nottingham, M. and Sayre, R. (2005): *The Atom Syndication Format*, <http://www.ietf.org/rfc/rfc4287.txt> (accessed Jan, 2007)

- Ohmukai, I. & Takeda, H. Semblog: Personal Knowledge Publishing Suite with Weblog. In proceedings of WWW 2004 Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics, New York, USA, 18 May 2004. <http://www-kasm.nii.ac.jp/papers/takeda/04/www2004ohmukai.pdf>
- Paquet, Sébastien; Pearson, Phillip (2004): A Topic Sharing Infrastructure for Weblog Networks, In: Proceedings of the Annual Conference on Communication Networks and Services Research (CNSR'04), IEEE Computer Society Press
- Paquet, Sébastien; Pearson, Phillip (2004): A Topic Sharing Infrastructure for Weblog Networks, Proceedings of the Second Annual Conference on Communication Networks and Services Research (CNSR'04), IEEE Computer Society Press
- Pchere (2005): One Click Multiple Blog Services Pinging Tools, <http://www.quickonlinetips.com/archives/2005/09/one-click-multiple-blog-services-pinging/>, last access: June 27th, 2007
- Pilgrim, Mark (2003): The Atom API, <http://www.xml.com/pub/a/2003/10/15/dive.html>, last access: July 5th, 2007
- Pilgrim, Mark (2004): The myth of RSS compatibility, <http://diveintomark.org/archives/2004/02/04/incompatible-rss>, last access: July 5th, 2007
- Radke, Andreas; Simmons, Brent; Winer, Dave (1999): Manila-RPC interface, <http://www.xmlrpc.com/manilaRpc>, last access: July 5th, 2007
- Reese, Byrne (2006): Standardizing the Atom Thread Extension, http://www.majordomo.com/atom/standardizing_the_atom_thread_extension.php (accessed Jan, 2007)
- Schiano, Diane; Nardi, Bonnie; Gumbrecht, Michelle; Swartz, Luke (2004): Blogging by the Rest of Us, In: Proceedings of the CHI 2004, Vienna, Austria, ACM Press, pp. 1143 – 1146
- Schmidt, Jan; Mayer, Florian (2007): Wer nutzt Weblogs für collaborative Lern- und Wissensprozesse?, In: Dittler, Kindt, Schwarz (Eds.): Online-Communities als Soziale Systeme, Waxmann, New York/München/Berlin
- Semantic Web Advanced Development – Europe, <http://www.w3.org/2001/sw/Europe/>
- Snell, J. (2006): Atom Threading Extensions, <http://www.ietf.org/rfc/rfc4685.txt> (accessed Jan, 2007)
- Technorati (2007): Technorati: About Us, <http://www.technorati.com/about/>, accessed: July 5th, 2007
- Tepper, Michele (2003): The Rise of Social Software, In: netWorker 7(3), ACM Press, pp. 18 – 23

Trafford, P., (2004), Remote Authoring of Mobile Blogs for Learning Environments (RAMBLE) project, http://ramble.oucs.ox.ac.uk/archive/bid/JISC_RAMBLE_final.htm (accessed Jan, 2007)

Trott, Ben; Trott, Mena (2002): TrackBack Technical Specification, Sixapart, http://www.sixapart.com/pronet/docs/trackback_spec, last access: July 5th, 2007

Van der Heijden, Kees (1997): Scenarios, Strategy, and the Strategy Process, In: presearch 1(1), Global Business Network, <http://www.gbn.com/ArticleDisplayServlet.srv?aid=550>, last access: July 5th, 2007

Wikipedia (2007): Authorization, <http://en.wikipedia.org/wiki/Authorization>, last-access: June 25th, 2007

Wild, Fridolin; Sobernig, Stefan (2006): Interoperability Framework Draft for the Distributed Open Virtual Learning Environment, Deliverable D3.1, iCamp Project

Williams, Evan (2001): Blogger API, http://www.blogger.com/developers/api/1_docs/, last access: June 25th, 2007

Winer, Dave (1999): XML-RPC Home Page, <http://www.xmlrpc.com/>, last access: July 5th, 2007

Winer, Dave (2000): OPML 1.0 Specification, <http://www.opml.org/spec>, last access: June 26th, 2007

Winer, Dave (2003): RSS 2.0 at Harvard Law, <http://cyber.law.harvard.edu/rss/rss.html>, last access: July 5th, 2007

Wittenbrink, Heinz (2005): Newsfeeds mit RSS und Atom, Galileo Press, Bonn

Wordpress (2007): Importing Content, http://codex.wordpress.org/Importing_Content, last access: July 5th, 2007

The iCamp Consortium.

Centre for Social Innovation (CSI)	Coordinator	Austria
Jozef Stefan Institute (JSI)	Contractor	Slovenia
University of Leicester (ULE)	Contractor	United Kingdom
Universidad Politécnic de Madrid (UPM)	Contractor	Spain
Vienna University of Economics and Business Administration (VUE)	Contractor	Austria
University of Science and Technology (AGH)	Contractor	Poland
Kaunas University of Technology (KTU)	Contractor	Lithuania
Isik University (ISIK)	Contractor	Turkey
Tallinn University (TLU)	Contractor	Estonia
Tomas Bata University in Zlín (TBU)	Contractor	Czech Republic

