Model for Collaborative Decision Making Based on RDF Reification

Dmitry Borodaenko

angdraug@debian.org

Abstract. This paper presents a novel approach to online collaboration on the Web, intended as technical means to make collective decisions in situations when consensus fails. It is proposed that participants of the process are allowed to create statements about site resources and, by the means of RDF reification, to assert personal approval of such statements. Arbitrary algorithms may then be used to determine validity of a statement in a given context from the set of approval statements by different participants. The paper goes on to discuss applicability of the proposed approach in the areas of open-source development and independent media, and describes its implementation in the Samizdat open publishing and collaboration system.

1 Introduction

Extensive growth of Internet over the last decades introduced a new form of human collaboration: online communities. Availability of cheap digital communication media has made it possible to form large distributed projects, bringing together participants who would be otherwise unable to cooperate.

As more and more projects go online and spread across the globe, it becomes apparent that new opportunities in remote cooperation also bring forth new challenges. As observed by Steven Talbott[9], technogical means do not provide a full substitute for a real person-to-person relations, “technology is not a community”. A well-known example of this is the fact that it is vital for an online communty to augment indirect and impersonal digital communications with live meetings. However, even regular live meetings do not solve all of the remote cooperation problems as they are limited in time and scope, and thus can’t happen often enough nor include all of the interested parties into communication. In particular, one of the problems of online communities that is begging for a new and better technical solution is decision making and dispute resolution.

While it is most common that online communities are formed by volunteers, their forms of governance are not necessarily democratic and vary widely, from primitive single-person leadership and meritocracy in less formal technical projects to consensus and majority voting in more complicated situations.

Usually, decision making in online volunteer projects is carried out via traditional communication means, such as IRC channels, mailing lists, newsgroups, etc., with rare exceptions such as the Debian project which employs its own

Devotee voting system based on PGP authentication and Concorde vote counting[6], and the Wikipedia project which relies on a Wiki collaborative publishing system and enforces consensus among its contributors. The scale and the level of quality achieved by the latter two projects demonstrates that formalized collaboration process is as important for volunteer projects as elsewhere: while sufficient to determine rough consensus, traditional communications require participants to come up with informal means of dispute resolution, making the whole process overly dependent on interpersonal attitudes and communicative skills within group.

It is not to say that Debian or Wikipedia processes are perfect and need not be improved. The strict consensus required by the Wikipedia Editors Policy discourages dissenting minority from participation, while full-scale voting system like Debian Devotee can't be used for every minor day-to-day decision because of the high overhead involved and the limits imposed by the ballot form.

This paper describes how RDF statement approval based on reification can be applied to the problem of online decision making in diverse and politically intensive distributed projects, and proposes a generic semantic model which can be used in a wide range of applications involving online collaboration. The proposed model is implemented in the Samizdat open-publishing and collaboration engine, described later in the paper.

2 Collaboration Model

The collaboration model implemented by Samizdat evolves around the concept of *open editing*[8], which includes the processes of publishing, structuring, and filtering online content. "Open" part of open editing implies that the collaboration process is visible to all participants, and roles of readers and editors are available equally to everyone. *Publishing* involves posting new documents, comments, and revised documents. *Structuring* involves categorization and appraisal of publications and other actions of fellow participants. *Filtering* process is intended to reduce information flow to a comprehensible level by presenting a user with resources of highest quality and relevance. Each of these processes requires a fair amount of decision making to be done, which means that its effectiveness can be greatly improved by automating some aspects of the decision making procedure.

3 Collective Statement Approval

3.1 Focus-Centered Site Structure

In the proposed collaboration model, RDF statements are used as a generic mechanism for structuring site content. While it is possible to make any kinds of statements about site resources, the most important kind of statement is the one that relates a resource to a so-called "focus"[2]. *Focus* is a kind of resource that, when related by an RDF statement to other resources, allows to group similar resources together and to evaluate resources against different criteria. In some sense, all activities of project members are represented as relations between resources and focuses.

Dynamically grouping resources around different focuses allows project members to concentrate on the resources that are most relevant to their area of interests and provide best quality. Use of RDF for site structure description makes it possible to store and exchange filters for site resource selection in the form of RDF queries, thus allowing participants to share their preferences and ensuring interoperability with RDF-aware agents.

Since any resource can be used as a focus, it is possible that project members define their own focuses, and relate focuses one to another. In a sufficiently large and intensive project, this feature should help site structure to evolve in accordance with usage patterns of different groups of users.

3.2 RDF Reification

RDF reification provides a mechanism for describing RDF statements. As defined in "RDF Semantics"[7], assertion of reification of RDF statement means that a document exists containing a triple token instantiating the statement. The reified triple is a resource which can be described in the same way as any other resource. It is important to note that there can be several triple tokens with the same subject, object, and predicate, and, according to RDF reification semantics, such tokens should be treated as separate resources, possibly with different composition or provenance information attached to each.

3.3 Proposition and Vote

In the proposed model, all statements are reified, and may be voted upon by project members. To distinguish statements with attached votes, they are called "propositions". *Proposition* is a subclass of RDF statement which can be approved or disapproved by votes of project members. Accordingly, *vote* is a record of vote cast in favor or against particular proposition by particular member, and *rating* is a denotation of approval of the proposition as determined from individual votes.

Exact mechanism of rating calculation can be determined by each site, or even each user, individually, according to average value of votes cast, level of trust existing between the user and particular voters, absolute number of votes cast, etc. Since individual votes are recorded in RDF and are available for later extraction, rating can be calculated at any time using any formula that suits the end user best. Some users may choose to share their view of the site resources, and publish their filters in the form of RDF queries.

Default rating system in Samizdat lets voter select from ratings "-2" (no), "-1" (not likely), "0" (uncertain), "1" (likely), "2" (yes). Total rating of proposition is equal to the average value of all votes cast for the proposition; resources with rating below "-1" are hidden from view.

4 Target Applications and Use Cases

4.1 Open Publishing

While it is vital for any project to come up with fair and predictable methods of decision making, it's hard to find a more typical example than the Indymedia network, international open publishing project with the aim of providing the public with unbiased news source[1]. Since the main focus of Indymedia is politics, and since it is explicitly open for everyone, independent media centers are used by people from all parts of political spectrum, and often become a place of heated debate, or even target of flood attacks.

This conflict between fairness and political bias, as well as sheer amount of information flowing through the news network, creates a need for a more flexible categorization and filtering system that would take the burden and responsibility of moderation off from site administrators. The issue of developing an open editing system was raised by Indymedia project participants in January 2002, but, to date, implementations of this concept are not ready for production use. The Active2 project[10] which has set forth to fulfil that role is still in the alpha stage of the development, and, unlike Samizdat, limits its use of RDF to describing its resources with Dublin Core meta-data.

Implementation of an open editing system was one of the initial goals of the Samizdat project[4], and deployment of the Samizdat engine by an independent media center would become a deciding trial of vitality of the proposed collaboration model in a real-world environment.

4.2 Documentation Development

Complexity level of modern computer systems makes it impossible to develop and operate them without extensive user and developer manuals which document intended behaviour of a system and describe solutions to typical user problems. Ultimately, such manuals reflect collective knowledge about a system, and may require input from many different people with different perspectives. On the other hand, in order to be useful to different people, documentation should be well-structured and easy to navigate.

The most popular solution for collaborative documentation development to date is *Wiki*, a combination of very simple hypertext markup and ability to edit documents within an HTML form. Such simplicity makes Wiki easy to use, but in the same time limits its applicability to large bodies of documentation. Due to being limited to basic hypertext without categorization and filtering capabilities, Wiki sites require huge amount of manual editing done by trusted maintainers in order to keep the site structure from falling behind a growing amount of available information, and to protect it from vandals. Although there are successful examples of large Wiki sites (most prominent being the Wikipedia project), Wiki does not provide sufficient infrastructure for development and maintainance of complex technical documentation.

Combination of the Wiki approach with RDF metadata, along with implementation of the proposed collaborative decision making model for determination of documentation structure, would allow to make significant progress in the adoption of the open-source software which is often suffering from a lack of comprehensive and up-to-date documentation.

4.3 Bug Tracking

Bug-tracking tools have grown to become essential component of any software development process. However, despite wide adoption, bug-tracking software has not yet reached maturity: interoperability between different tools is missing; incompatible issue classifications and work flows complicate status syncronization between companies collaborating on a single project; lack of integration with time-management, document management, version control and other kinds of applications increases amount of routine work done by project manager.

On the other hand, development of integrated project management systems shows that the most important problem in project management automation is convergence of information from all sources in a single focal point. For such convergence to become possible, unified process flow model, based on open standards such as RDF, should be adopted across all information sources, from source code version control to developer forums. Since strict provenance tracking is a key requirement for such model, the proposed reification-based approach may be employed to satisfy it.

5 Samizdat Engine

5.1 Project Status

Samizdat engine is implemented in the Ruby programming language and relies on the PostgreSQL database management system for RDF storage. Other programs required for Samizdat deployment are Ruby/Postgres, Ruby/DBI, and YAML4R libraries for Ruby, and Apache web server with mod_ruby module. Samizdat is free software and does not require any non-free software to run[5].

Samizdat project development started in December 2002, first public release was announced in June 2003. As of the second beta version 0.5.1, released in March 2004, Samizdat provided basic set of open publishing functionality, including registering site members, publishing and replying to messages, uploading multimedia messages, voting on relation of site focuses to resources, creating and managing new focuses, hand-editing or using GUI for constructing and publishing Squish queries that can be used to search and filter site resources. Next major release 0.6.0 is expected to add collaborative documentation development functionality.

5.2 Samizdat Schema

Core representation of Samizdat content is RDF. Any new resource published on Samizdat site is automatically assigned a unique numberic ID, which, when appended to the base site URL, forms resource URIref. This ID may be accessed via `id` property. Publication time stamp is recorded in `dc:date` property (here and below, "`dc:`" prefix refers to the Dublin Core namespace):

```
:id
        rdfs:domain rdfs:Resource .

dc:date
        rdfs:domain rdfs:Resource .
```

`Member` is a registered user of a Samizdat site (synonyms: poster, visitor, reader, author, creator). Members can post messages, create focuses, relate messages to focuses, vote on relations, view messages, use and publish filters based on relations between messages and focuses.

```
:Member
        rdfs:subClassOf rdfs:Resource .

:login
        rdfs:domain :Member ;
        rdfs:range rdfs:Literal .
```

Resources are related to focuses with `dc:relation` property:

```
:Focus
        rdfs:subClassOf rdfs:Resource .

dc:relation
        rdfs:domain rdfs:Resource ;
        rdfs:range :Focus .
```

`Proposition` is an RDF statement with `rating` property. Value of `rating` is calculated from `voteRating` values of individual `Vote` resources attached to this proposition via `voteProposition` property:

```
:Proposition
        rdfs:subClassOf rdf:Statement .

:rating
        rdfs:domain :Proposition ;
        rdfs:range rdfs:Literal .

:Vote
        rdfs:subClassOf rdfs:Resource .

:voteProposition
        rdfs:domain :Vote ;
        rdfs:range :Proposition .

:voteMember
        rdfs:domain :Vote ;
        rdfs:range :Member .

:voteRating
        rdfs:domain :Vote ;
        rdfs:range rdfs:Literal .
```

Parts of Samizdat schema that are not relevant to the discussed collective decision making model, such as discussion threads, version control, and aggregate messages, were omitted. Full Samizdat schema in N3 notation can be found in Samizdat source code package.

5.3 RDF Storage Implementation

To address scalability concerns, Samizdat extends traditional relational representation of RDF as a table of {subject, object, predicate} triples with a unique RDF-to-relational query translation technology. Most highly used RDF properties of Samizdat schema are mapped into fields of *internal resource tables* corresponding to resource classes, with id of the record referencing to the `Resource` table; all other properties are recorded as triples in the `Statement` table. Detailed explanation of the RDF-to-relational mapping can be found in "Samizdat RDF Storage"[3] document.

To demonstrate usage of the Samizdat RDF schema described earlier in this section, the exerpt of Ruby code responsible for individual vote rating assignment is quoted below.

```
def rating=(value)
    value = Focus.validate_rating(value)
    if value then
        rdf.assert %{
UPDATE ?rating = '#{value}'
WHERE (rdf::subject ?stmt #{resource.id})
      (rdf::predicate ?stmt dc::relation)
      (rdf::object ?stmt #{@id})
      (s::voteProposition ?vote ?stmt)
      (s::voteMember ?vote #{session.id})
      (s::voteRating ?vote ?rating)
USING PRESET NS}
        @rating = nil   # invalidate rating cache
    end
end
```

In this attribute assignment method of `Focus` class, RDF assertion is recorded in extended Squish syntax and populated with variables storing the rating `value`, resource identifier `resource.id`, focus identifier `@id`, and identifier of registered member `session.id`. When the Samizdat RDF storage layer updates `Vote.voteRating`, average value of corresponding `Proposition.rating` is recalculated by a stored procedure.

6 Conclusions

Initially started as an RDF-based open-publishing engine, Samizdat project opens a new approach to online collaboration in general. Proposed model of collective statement approval via RDF reification is applicable in a large range of problem domains, including documentation development and bug tracking.

Implementation of the proposed model in the Samizdat engine proves viability of RDF not only as a metadata interchange format, but also as a data model that may be employed by software architects in innovative ways. Key role played by RDF reification in the described model shows that this comparatively obscure part of RDF standard deserves broader mindshare among Semantic Web developers.

References

1. Arnison, Matthew: Open publishing is the same as free software, 2002 http://www.cat.org.au/maffew/cat/openpub.html
2. Borodaenko, Dmitry: Samizdat Concepts, December 2002 http://savannah.nongnu.org/cgi-bin/viewcvs/samizdat/samizdat/doc/concepts.txt
3. Borodaenko, Dmitry: Samizdat RDF Storage, December 2002 http://savannah.nongnu.org/cgi-bin/viewcvs/samizdat/samizdat/doc/rdf-storage.txt
4. Borodaenko, Dmitry: Samizdat — RDF model for an open publishing and cooperation engine. Third International OSCOM Conference, Berkman Center for Internet and Society, Harvard Law School, May 2003 http://slideml.bitflux.ch/files/slidesets/503/title.html
5. Borodaenko, Dmitry: Samizdat RDF Implementation Report, September 2003 http://lists.w3.org/Archives/Public/www-rdf-interest/2003Sep/0043.html
6. Debian Constitution. Debian Project, 1999 http://www.debian.org/devel/constitution
7. Hayes, Patrick: RDF Semantics. W3C, February 2004 http://www.w3.org/TR/rdf-mt
8. Jay, Dru: Three Proposals for Open Publishing — Towards a transparent, collaborative editorial framework, 2002 http://dru.ca/imc/open_pub.html
9. Talbott, Stephen L.: The Future Does Not Compute. O'Reilly & Associates, 1995 http://www.oreilly.com/~stevet/fdnc/
10. Warren, Mike: Active2 Design. Indymedia, 2002. http://docs.indymedia.org/view/Devel/DesignDocument