Virtual Communities as Narrative Processes

Marco Benini and Federico Gobbo

Dipartimento di Informatica e Comunicazione
Università degli Studi dell'Insubria
via Mazzini 5, IT-21100, Varese, Italy
{marco.benini, federico.gobbo}@uninsubria.it

Summary. By facing the problem to describe the history of a virtual community as the sequence of events generated by its participants, a different perception of the meaning of communitywares emerges. This paper describes a proposal for a virtual community system based on the narrative process that supports the social evolution of the community.

Key words: Communityware design, Semantic web technology, Ontologies

1 Introduction

Discussion and collaboration servers, i.e. software tools that promote and mediate dialogue and partnership among different users, are not a novelty anymore: they gradually grew following the network spread and evolution [1]. Nevertheless, there is still no tool supporting the evolution of the rules of the social network underpinning cooperation. In fact, the social rules are mostly defined by designers, and hard-coded into the collaborative system, without explicit semantic information. In this paper a formal model to represent the most known collaboration models is given, in terms of semantic web ontologies. These ontologies will be managed in natural language, so that users can define by themselves new, original forms of cooperation and dialogue.

In principle, there are three basic collaboration models over a network: e-mail exchange (including mailing lists), shared repositories, and interactive content update technologies [7]. Virtual communities, encouraging participation and active learning among remote users, naturally prefer the third model since their members aim to establish social relations, and this goal is easier to achieve if users are allowed to update content interactively.

As a matter of fact, virtual communities evolved around complex communitywares which combine the feature of the three basic models. In fact, their main service [1] was to provide discussion lists, often called *conferences*, where the participants were allowed to discuss over common topics (the e-mail exchange model), while additional features were usually provided, as shared repositories (i.e. the second model), or personal web pages, email address, etc. The aim behind these systems was to offer an all-inclusive environment [11], in order to give a complete support to each participant's need, so that the community members were invited to use the Internet almost exclusively through the community support.

Henceforth, as communities evolved, the software platforms grew in complexity, due to the subsequent addition of unplanned features. In fact, it is very difficult, if not impossible, to foresee every participant's need or desire in advance, i.e. before the virtual community establishes itself, as people expectations are usually very different: our claim is that these wishes cannot be foreseen since they arise **after** the community uses the software for enough time to evolve itself, while the design of the software takes place **before** the community starts to operate.

2 Virtual communities and "new texts"

Since the end of the 20th century, the increase of network size and speed and the standardisation of the web [2] also lead to a deep transformation of virtual communities. Community services became differentiated according to their needs. In our opinion, the deep reasons behind this fragmentation lie in the increase of users' awareness: the Internet services are now mostly well-known and, thus, users don't need an active guidance in their usage anymore. Indeed, it is not surprising the raise in popularity of a new kind of community-oriented services, broadly called *new texts*, like for instance *wikies*, which allow the collaborative development of knowledge, or *blogs*, which act as discussion vehicles [5]. However, despite their maturity as technological objects, the design of communitywares and new text services is similar and still quite traditional: they are usually developed as specialised web-based applications [11].

Their design and development is focused on the web technology and its clever application to the problem domain; the simple idea that the purpose of the software is **just** to support a living community is left in the background. In the approach proposed here, the reversal is true: a communityware should support a virtual community from its start permitting its evolution with the social rules that participants arbitrarily decide to adopt, according to the community life. Moreover, the social rules belong to the community, which can modify them over time to reflect new needs and wishes.

2.1 Blogs and wikies as narratives

Since our aim is to propose a reversal approach, in the following we will describe ideal "new text" communitywares and how their core works. A designer may either directly implement this approach, or, preferably, may include techniques and ideas in a richer system, where the features avoided for clearness and conciseness are present and fully supported.

We start by designing and thus constructing a language allowing the writing of the community history. Hence we call our approach *narrative*, since virtual communities are considered as narrative processes. This narration is described by means of a language, which has enough expressive power to depict also the community state, that is, the information owned by the community. In this perspective, the language itself is part of the state; since the state varies over time, and the language is part of it, the language may evolve as well. As far as the features used by the community are defined by the language, any addition to the language corresponds to an evolution of the community in terms of represented features, thus overcoming the discussed ageing problem.

In order to concretely exemplify our ideas we define the words “User”, “Message” and “Conference”. Their intended meaning[1] is as follows: the users are the actors of the community, i.e. they can perform actions like sending messages, and, in turn, messages are organised to form conferences. The community state is the sum of the conferences and the language defined insofar. The community history tales the changes in the community state.

Communitywares based on the e-mail exchange model [7] – e.g. BBS, mailing lists, web forums, web groups – organise content on the paradigm “write once, read many”. In fact, in this paradigm, conferences are *threads*, owned by no user in particular. A message, the *root*, starts a thread on a specific question or topic, which sequentially people answer or comment. If a message is off-topic, a new thread begins. Threads are often very long, and the result is a complex tree of messages, where conference boundaries are not always clear as messages belong to more than one conferences, and redundancy in the messages content is tolerated [7].

Blogs are a significant variant of this paradigm, which we call the *annotation model.* In fact, unlike what happens with mailing lists, blogs have a clearly defined author – maybe collective, but still one – who owns the conferences and has the right to manage their messages. Conferences are shaped as threads, but the root (called *post* in the blog jargon) is more important than the threaded answers, which can be considered as mere comments. Unlike mailing lists, threads are usually short, and not rarely they are made by a unique message, the post. Comments are not the only way to answer to one’s post: blogs are by no means living as monads, on the contrary, *annotations* – i.e. messages belonging to a blog but pertaining to another blog post – are allowed and encouraged. When annotation happens, blogs are put into relation and form a *blogosphere* – another form of community [10]. Fig. 1 shows a prototypical example: John has raised an issue (post B) for further considerations on Tuesday in his blog, and Pietro reacts commenting it in John’s blog space. On the contrary, Mario, after reading B, decides to write the longer answer D as an annotation of B, perhaps via a citation. Thus, on Wednesday John’s and Mario’s blogs are intertwined. In this picture, Jack is allowed to comment but he decides to read without reacting. In blogs, the

[1] These notions are standard and described at length, e.g., in [1].

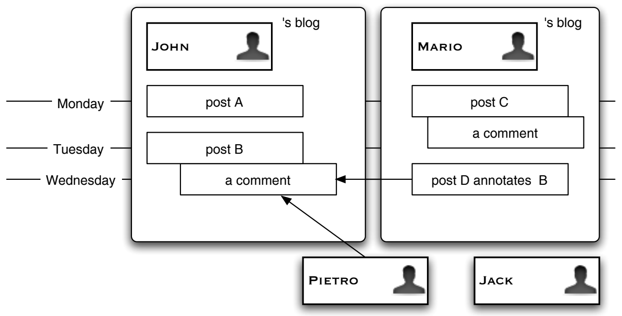

Fig. 1. A minimal blogosphere

content is organised on the paradigm "write yours, read and comment the others". In the terms given above, a blog is a set of conferences owned by a user with a defined identity. Wikies are quite different from blogs, as in the

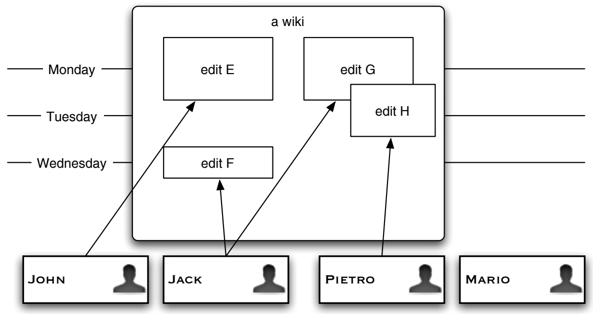

Fig. 2. A pure wiki

shown example (Fig. 2). On Monday John and Jack start two different wiki pages (edit E, G) on some related topics – in our terms, they had created two conferences. Note that in a pure wiki no message is authored, i.e. all messages are anonymous. Pietro reads Jack's message on Tuesday, and he writes a message (edit H) that updates the message body. Jack, who evidently likes the wiki way more than blogs, reads the changes and decides to add some information to the conference started by John, appending some content (edit F). The conference history becomes a sequence of patches of differences between subsequent messages.

3 From natural language to ontologies

When describing a blog or a wiki, it is natural and easier to depict an example of the intended model, as done in the previous section.

One can try to move toward a formal description by *narrating* the example: "John is an user. John's blog is a set of conferences, owned by John. A comment is a message. Only users may post messages". This informal description identifies some social rules, some entities and some roles: John and his blog are entities, they gain a social role by their attributes, like being a user or a blog. There are other, unidentified entities, like messages, and they can be related to known objects by some actions, like "post", whose usage is restricted to users when involving comments.

In this perspective, narrating the history of the community means to record the sequence of actions performed in the community world. Every action is composed by a series of *events*, each one described by a simple sentence. When able to interpret the meaning of events as actions to apply on the actual community information, the community will be enlivened by a suitable engine that receives and performs the action on the community state. If narratives, e.g. the example before, are formalised into events, it becomes feasible to develop an engine for a narrative communityware.

Our investigation on the informal description starts by analysing the sentences: sentences are structured groups of elements, where each element plays a role or a defined function – hierarchically defined. According to Tesnière's structural grammars [12], a sentence is a set of connections, where its type is defined by the verb: the term *valence* refers to the number of arguments, or *actants*, a verb can take. For example, the sentence "John owns John's Blog" contains "owns" which is a divalent verb, i.e. it has a first actant ("John", the subject) and a second actant. On the contrary, the verb "to be" is monovalent, as it has only one actant (the subject), and denotes an attribute of it. In order to avoid unnecessary complexity in natural language parsing, only present tense is used, i.e. sentences are all statements. Besides verbs, there are nouns: generally speaking they denote either concrete entities, like "John", or concepts, like "user". Some verbs and some nouns are predefined, i.e., their meaning is common knowledge. These elements are "to be" and "may" in our example. On the contrary, most nouns and verbs have a specific meaning which depends on the particular community we are constructing: "user", "conference", "to own" and "to post" are of this kind, since their interpretation varies if the community is a wiki, a blog or something else. Therefore, a formal description must define these notions and a communityware engine has to provide a model, enabling their subsequent use on the community state.

The description we propose is based on a pair of knowledge bases, represented as OWL ontologies: the *history* and the *state* of the community. The history contains the recording of the sequence of events occurring during the community life. The state contains the language definitions and the information owned by the community as it holds in a particular instant; while the history is constantly growing, the state gets updated by every event. In this respect, using an ontology to represent the state allows both to dress the language with a logical meaning and to ensure the formal consistency of the depicted community world moment by moment.

3.1 Sketches from a Narrative Community

The narrative approach can be formalised in an operative model of the previous examples: we start by defining a simple language that allows the narration of the community events; the events will be the actions each participant performs in the community. The syntax is based on a set of nouns and verbs that allows the constructions of simple sentences: for convenience, we use the OWL syntax [4, 8][2] that simplifies the understanding of the system's behaviour.

In the beginning, the history of the community as well as its state are empty, and the language is pure OWL plus the `vcs` (*virtual community structure*) namespace, whose content is explained later. The first step is to define the notions of "User", "Message" and "Conference". A user, the community starter, narrates the following events to the system:

```
<owl:Class rdf:ID="Noun" />
<owl:Class rdf:ID="User">
  <rdfs:subClassOf rdf:resource="#Noun" />
</owl:Class>
<owl:Class rdf:ID="Message" />
  <rdfs:subClassOf rdf:resource="#Noun" />
</owl:Class>
<owl:Class rdf:ID="Conference" />
  <rdfs:subClassOf rdf:resource="#Noun" />
</owl:Class>
```

A "Noun" is rendered as an OWL class; "User", "Message" and "Conference" are nouns. Analogously, he can describe the basic verbs to interact with the concepts just defined:

```
<owl:Class rdf:ID="Verb">
  <rdfs:subClassOf rdf:resource="&owl:ObjectProperty" />
</owl:Class>
<Verb rdf:ID="read">
  <rdfs:domain rdf:resource="#User" />
  <rdfs:range>
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#Message" />
      <owl:Class rdf:about="#Conference" />
    </owl:unionOf>
  </rdfs:range>
  <vcs:action> ... </vcs:action>
```

[2] We assume the standard conventions for namespaces in OWL fragments, see [8].

```
</Verb>
<Verb rdf:ID="own">
  <rdfs:domain rdf:resource="#User" />
  <rdfs:range rdf:resource="#Conference" />
  <vcs:action> ... </vcs:action>
</Verb>
<Verb rdf:ID="post">
  <rdfs:domain rdf:resource="#User" />
  <rdfs:range rdf:resource="#Message" />
  <vcs:action> ... </vcs:action>
</Verb>
```

Therefore, a verb like “read” is both a linguistic element in the “Verb” class, and an OWL-property whose domain (the subject of the verb) is a “User” and whose range (the object of the verb) is either a “Message” or a “Conference”. Consequently, “read” has a triple meaning: as a linguistic element, it is a bivalent verb; as an action, it denotes the transformation on the state as calculated by its `<vcs:action>` tag; as an OWL element, it is a property relating class elements. In particular, the `<vcs:action>` declares the effect of the verb on the state of the community by means of a program written in XML/XQuery [3], linked via the `vcs` namespace – details are omitted here for clarity. The state of the community is an OWL ontology containing the sentences in the defined language that describe the information of the community. The action is a piece of programming code that defines the change on the state when a related event happens: for example, when the event “John posts the message X” occurs, the message X is added to the community state by the program contained in the `<vcs:action>` of the declaration of the “post” verb (see next subsection for a precise description).

To describe the structure of a message, we enrich our language with attributes, represented as OWL datatype properties:

```
<owl:DatatypeProperty rdf:ID="title">
  <rdfs:domain rdf:resource="#Message" />
  <rdfs:range rdf:resource="&xsd:string" />
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="content">
  <rdfs:domain rdf:resource="#Message" />
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="inConference">
  <rdfs:domain rdf:resource="#Message" />
  <rdfs:range rdf:resource="#Conference" />
</owl:ObjectProperty>
```

To show how the previous declarations can be used, we populate the state with some facts, from the examples in Sec. 2.1:

```
<User rdf:ID="John" />
<Conference rdf:ID="JohnBlog" />
```

```
<Message rdf:ID="msg1">
  <title> Post A </title>
  <content rdf:resource="http://www.dicom.uninsubria.it" />
  <inConference rdf:resource="#JohnBlog"/>
</Message>
<User rdf:about="#John">
  <own rdf:resource="#msg1" />
</User>
```

The narrative approach, as described till now, allows both to write the history of the community, and to operate the core actions on the community state. Moreover, the language used to tale the events is defined as part of the narration, like in mathematical textbooks, where the concepts are first defined, and then used to derive results and to define new notions.

In the emerging model, nothing prevents the reflective usage of already defined concepts. For example, we can define a conference whose elements are the defined users. The event we submit to the system is the following:

```
<Conference rdf:ID="Users" />
<owl:Class rdf:about="#User">
  <rdfs:subClassOf rdf:resource="#Message" />
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#inConference" />
      <owl:allValuesFrom rdf:resource="#Users" />
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```

It means that a user is a special kind of “Message”, which lies in the “Users” conference. The result is that user management does not require new verbs or special actions: the ability to post in the “Users” conference allows the creation, cancellation and modification of the set of users by means of the very same actions used to manage any other message. An important point to notice is that we **evolved** toward this kind of management: in fact, we incrementally derived the idea of managing the users adding a new conference to an existing community where, initially, “User” was a standalone concept.

The reflective use of concepts is nothing but an example of evolution: in fact, since the language may be modified at any time, potentially every event involving a change in the language can be regarded as a step toward the evolution of the community.

4 Behind the Curtain

The ideal communityware engine implicitly used in the preceding section is very simple: it takes its input, an *event*, from the web, processes it and calculates the output, usually an XML document. The event is an OWL fragment, that must be understandable in the current state, that is, the state ontology plus the event must form a valid XML document as defined in [13], satisfying the OWL syntax augmented with the `vcs` namespace.

Moreover, the event must be semantically sound with the state, that is, the state ontology plus the event must form an OWL-consistent document as defined in [9], thus generating a logically sound theory.

If the event is both valid and sound, it denotes the actions that must be performed on the ontology state: each action is defined by means of a function written in XML/XQuery [3], represented inside the definition of the simple event's verb via the `<vcs:action>` tag; the default action (when the tag is missing) is to append the event to the ontology state. Therefore, the denoted action is tentatively performed and the resulting state is checked to be valid and sound. As a consequence, the output is calculated as the updated state ontology: in a real system, this would be inappropriate and a suitable presentation of part of the state should be extracted and shown to the user. Finally, the event is recorded in the community history.

Although heavily based on the semantic web technologies, the described engine operates as a simplified web-based application. But, differently from the traditional communitywares, wikies and blogs, it does not provide hard-coded notions, actions and rules. As previously illustrated, even the basic notions, like *user* or *message*, are defined "in the language" and, thus, they become part of the state hence, as any other element of the state, they may be modified, created or cancelled with the only limitation that the resulting state preserves validity and soundness, i.e. it has to be a well-formed OWL ontology with no internal contradictions.

As a matter of fact, the abstractness and the generality of the illustrated engine provide the community with the instruments to sustain its own evolution, since literally everything can be discussed and, eventually, modified. It is evident that, in practice, a more significant starting point is needed, that is, the initial language should be non-empty and should represent a well recognised language to describe a community model. In this respect, the shown sketch is too limited, but the discussion in Sec. 2.1 provide the highlights to develop the notions and, thus, the language constructors needed to represent the corresponding community models, starting from blogs and wikies.

In fact, the narration of an example of community life requires a language that can be usefully represented in the form of an OWL ontology; this ontology becomes the foundational event of the community, enabling its usage by means of the illustrated engine. Therefore, the narrative description of communities becomes the enabling metaphor that allows their representation in a semantic web system. Because of the expressive power of semantic web conceptual instruments, it is possible to enliven the narrative representations of communities in order to support them and, eventually, their evolutions.

5 Concluding remarks

This paper has shown the idea that considering virtual communities as the result of a narrative process, leads to a new possible design approach of the communitywares to support them. This approach wants to suggest that the semantic web technology is mature enough to permit a significant encoding of virtual communities in its main representation language, namely OWL. In this respect, ontologies become the instrument both to represent and to operate communities, with a degree of freedom and flexibility unachievable in traditional and modern communitywares.

Being a proposal paper, a great deal of future work is expected: in the first place, the implementation of the engine and the consequent collection of experimental data. Also, it is important to study to what extent reflection (see the end of Sec. 3.1) can be used to simplify the management of complex communities. Finally, the proposed approach allows to simulate communities with specific social rules: for example, the study of the application of the rules formalising Creative Commons licenses [6]. Although we have begun to explore some of these themes [5], most is still to be done.

References

1. M. Benini, F. De Cindio, and L. Sonnante. Virtuose, a VIRTual CommUnity Open Source Engine for integrating civic networks and digital cities. In P. van den Besselaar and S. Koizumi, eds, *Digital Cities III — Information Technologies for Social Capital: Cross-Cultural Perspectives*, volume 3081 of *LNCS*, pp. 217–232. Springer Verlag, 2003.
2. T. Berners-Lee. *Weaving the Web.* Harper, 2002.
3. D. Chamberlin et al. *XQuery from the Experts.* Addison Wesley, 2003.
4. M. Dean and G. Schreiber. OWL web ontology language reference. W3C, Feb. 2004.
5. F. Gobbo, M. Chinosi, and M. Pepe. Novelle, a collaborative open source writing software. In J. Karlgren, ed., *NEW TEXT: Wikis and blogs and other dynamic text sources*, Trento, Italy, 2006. Association for Computational Linguistics.
6. L. Lessig. *Free Culture.* Penguin, 2004.
7. B. Leuf and W. Cunningham. *The Wiki Way: Quick Collaboration on the Web.* Addison Wesley, 2002.
8. D.L. McGuinness and F. van Harmelen. OWL web ontology language overview. W3C, Feb. 2004.
9. P.F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL web ontology language semantics and abstract syntax. W3C, Feb. 2004.
10. Andrew Rosenbloom. The blogosphere. *Comm. of the ACM*, 47(12), Dec. 2004.
11. D. Schuler. New communities and new community networks. In M. Gurstein, ed., *Community Informatics: Enabling Communities with Information and Communications Technologies*, Hershey, USA, 2000. Idea Publishing Group.
12. L. Tesnière. *Éléments de syntaxe structurale.* Klincksieck, Paris, 1959.
13. F. Yergeau et al. Extensible markup language (XML) 1.1. W3C, Feb. 2004.