VIRTUOSE: a Virtual Community Open Source Engine

M. Benini, Univ. of Insubria (benini@dsi.unimi.it)
F. De Cindio, Univ. of Milano (fiorella.decindio@unimi.it)
L. Sonnante, The Milano Community Network Foundation (leonardo.sonnante@rcm.inet.it)

Introduction

Despite the wish that citizens are co-designers of the Networked Society, it is very difficult to pursue and achieve this goal. Web sites and portals are set up by public and private bodies and designed by computer professionals hired to *inform* people and *offer* them *on-line services.*

On the contrary, we believe that:

- the possibility for citizens to participate in shaping the Networked Society relies also in influencing – through comments, remarks, suggestions and protests - the way in which on-line services are conceived, designed and implemented;
- the design of appropriate technology is a fundamental element to give people such a possibility, and, following [Ray00], a community must grow also by developing the technological tools;
- the web still suffers of its origin, i.e., to be a publishing - and therefore a broadcasting - application. To develop *on-line interactive services,* the web browsers have been compelled to become the standard interface for databases and database applications. Peer-to-peer communication occurs via email, mailing list, newsgroups, chats and co-operative applications: they are the basis for the development of a virtual *community*, but unfortunately the web-based versions of the above applications are quite often less natural and effective than the original ones.

In current web site and portals, this last area is usually disjoint from the other two ones, while it would be useful to understand that a community of citizens, through their conversations, enriches information and assesses services. Notice that this community-generated knowledge is different from and much richer than the content which can be provided by a single person. In this way, citizens would not be mere *users* of the information services or *customers* of the interactive services; they would become *co-designers of the Networked Society.*

Let's explain this concept through an example: let's suppose a citizen asks the list of movies in Milan this evening. By accessing the local forum "Cinema" s/he enriches this information with community-knowledge created through discussions, for instance: "beautiful plot". At this point, s/he would book a place in the closest hall giving the movie. After assisting the movie, s/he comes back and comments in the local forum the movie or the state of the cinema hall (e.g., from the point of view of the facilities for disabled people). Other citizens either confirm or question the remarks.

The same pattern can be applied in many situations, from the choice of home-banking service to the need of a laboratory for clinic analysis. In the last case, comments after service use provides a powerful way of assessing the quality of services.

As far as we know, no one of the existing software completely matches the above requirement: the CSuite software [Mcm98], does not match our needs because of its character-based user interface; the so called *application servers*, as for instance Midgard [MDG], are basically *publishing* oriented. More integrated solutions, such as Vignette, [Vig], are proprietary and expensive. To develop and implements our view of community, VIRTUOSE, the VIRTual CommUnity Open

Source Engine keeps the *message* as the fundamental entity which constitutes both the unit of information to be published and the unit of discussion among people.

Our proposed solution: VIRTUOSE

Virtuose has been developed in the framework of the project “Development of services and technologies to turn local communities to account in the Information Society" funded by the Lombardy Regional Government and carried on by AIReC (the Association for Informatics and Community Networks). In particular the software specification and implementation is a co-operation among the University of Milan, the University of Insubria, the Foundation RCM (i.e., the Participatory Foundation which holds and manages the Milan Community Newtork) and Pumpkin s.r.l., a small software factory risen from the University labs.

The distinguished feature Virtuose includes is to homogeneously couple both the publishing of information, and facilities for discussion, which characterise communication within communities. Facilities for the integration with other on-line services are not considered in this first release.

Key entities

In the design phase, the following key entities have been outlined as characterising a virtual community: conferences, views, messages, users, and profiles.

Conferences can be defined as information containers regarding a specific topic. This information is partitioned in messages, that is, a set of elements belonging to the same structure. The structure therefore defines the type of the message. Every conference has a set of accepted message types. This wide conception of conference and message, as implemented inside VIRTUOSE, allows community members, or the users if you prefer, to create and publish structured contents, but also claims for a tool for filtering. This is the function carried out by views.

Views establish the way in which users and conferences interact. A conference can only be accessed through a views. Specifically, a view contains:

- a set of message types; it’s a subset of message types accepted by the conference, the ones which can be accessed through the view;
- a visualisation layout;
- a set of permissions, i.e. actions that one is allowed to perform on the messages of that conference when accessed through the view.

Messages are the containers for the information belonging to the system and are stored in a structured manner. Each message has a structure that defines a set of fields that constitute the type of the message. The values of the fields are displayed by means of display functions; Permissions of view dictate actions that can be played on messages.

Users are the authors of messages and they are identified and classified by means of their role(s). Roles are modelled by the more general concept of*profile*, which defines the relationship between users and views. So profiles can identify a role or an homogeneous class of users.

System features

The basic system features have been implemented using the entities defined in the previous section.

According to the previous section’ definitions, *conferencing* means all the features concerning content creation by users that can be achieved by sending messages to conferences.

Thanks to views every user can play actions only on messages of the types listed in the view, and permissions set what actions s/he can play.

This kind of conference management offers the following advantages:

- it is possible to set up private conferences, that is conferences which only a subset of users can access, or conferences where few users can send messages to and the others can read published messages;
- it is possible to set up “publishing views”, that is views that show only messages with a structure suited for information publishing;
- setting up a “discussion view” for the same conference, it is possible to make available comments sent by users about the above mentioned messages; according to the above definition, comments are simply messages with a structure where there is a text field;
- users can access different types of messages according to their role (profile);
- users can play different actions on the different types of messages according to their role (profile); in fact, we believe that a role is defined by the actions that a user can take, and by the message types s/he is allowed to manipulate.

To make this conference management versatile enough, it is necessary that the administrator can easily create new types of messages and new views. So a virtual community administrator has an enhanced interface which allows the undertaking of the tasks of creating and maintaining users, profiles, views and conferences.

Others main system features can be summarised as follows.

- Registration and users management: all users are registered to achieve member recognition in the community.
- Profiles: they are used to link a user to a set of views, to identify a role or to group users in an homogeneous class.
- Searching features.
- Hosting (this word means that every user can create and manage her/his Web site on our virtual community server), private mail and calendar features. With regard to these features, it is important to point out the following two aspects: these features (that are very common on Web site managing virtual communities) are completely integrated in the system; mailbox, calendar and Web site are in fact implemented by special conferences with suited message types, views and permissions.

3. Architecture of the System

The architecture of Virtuose consists of a hierarchy of abstract machines which starts from a database management system and culminates in a graphical user interface for a Web browser.

In fact, in order to achieve the goal of having a community software with the required degree of flexibility as well as a well engineered piece of software, the system has been structured as a stack of virtual machines, each one with its internal style, language, interface and set of notions.

Specifically, the lower layer is the database, where community pieces of information are stored. On this level information is equivalent to data, the programming style is traditional imperative coding and the basic notions (conferences, messages, etc.) have been described according to a slight extension of the entity-relationship paradigm.

The next layer is the *Model*, that implements a semantic for the data stored in the database. Its approach is object-oriented, and the basic entities are modelled as classes. Its fundamental purpose is to shift the vision from an absolute to a relative one: in fact, in the database layer, the code can access every piece of data, and the code works in an environment which presents the community as a whole, while, in the model layer, the code acts on the data which can be accessed according to the user's permissions, and classes look at the system from the point of view of the user who executes them. So a concept of locality gets introduced, and the idea of role, as previously explained, gets established from the very roots of the VIRTUOSE system.

The third layer has the purpose of managing I/O with the web environment, thus modelling the idea of client-side event and the idea of HTML template and widget. The fourth and last layer provides the user interface to the system, by processing inputs from the user and producing the appropriate answers in the form of dynamic HTML pages.

The basic notions of conferences, messages, profiles, users, views, etc. are modelled in a different way at every layer in the architecture, and this rich and complex way of rendering the concepts, is the technical key which enabled us to achieve the goal under the slogan:

community knowledge[1] = publishing + debating.

4. References

[CS] CSuite Web site: http://csuite.ns.ca

[KN01] Kuutti, K., Nakata, K., *Second ECSCW Workshop on Community Knowledge*, Seventh European Conference on Computer Supported Cooperative Work, Bonn, September 2001.

[Mcm98] McMahon S., *Open Source Tools for Community Networks*, Cisler S. (ed.), "Technology Issue", Community Networking, quarterly of Assoc. For Community Networking, December, 1998.

[MDG] Midgard Project Web site: http://www.midgard-project.org

[Ray00] Raymond E., *The Cathedral and the Bazaar*, 2000

[Vig] Vignette Corp. Web site: http://www.vignette.com

[1] See also [KN01].